

FIG. 1
(Prior Art)

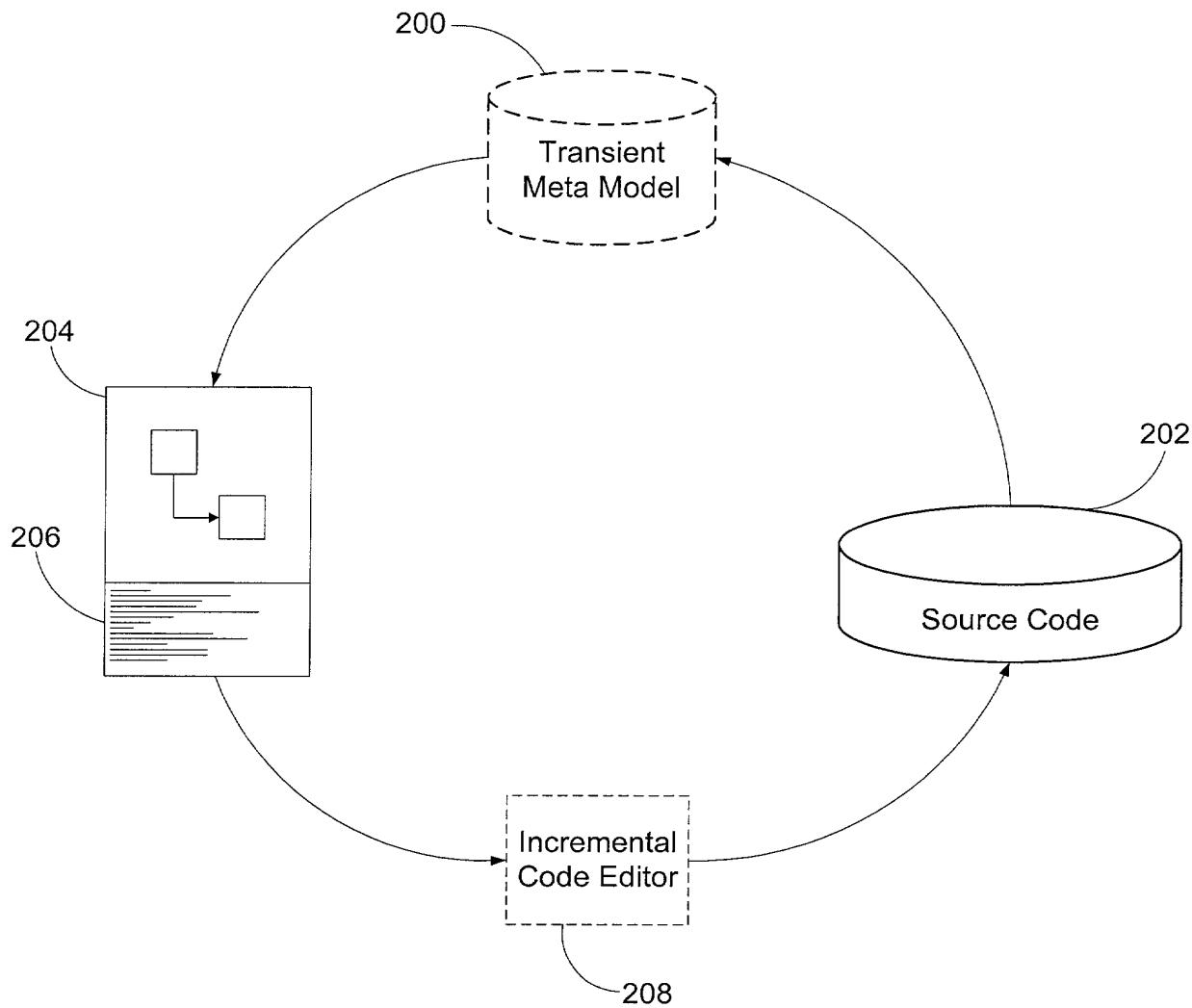


FIG. 2

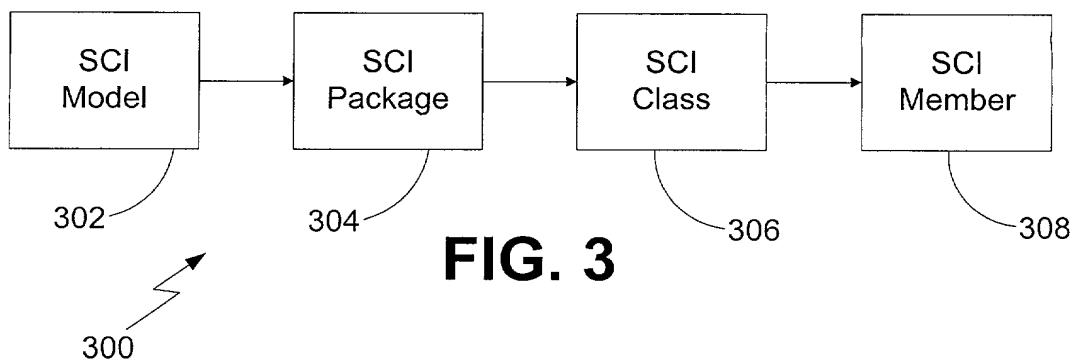


FIG. 3

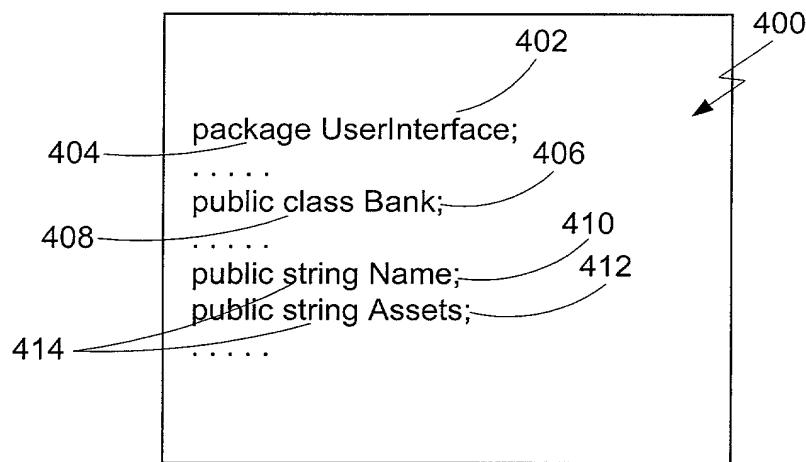


FIG. 4

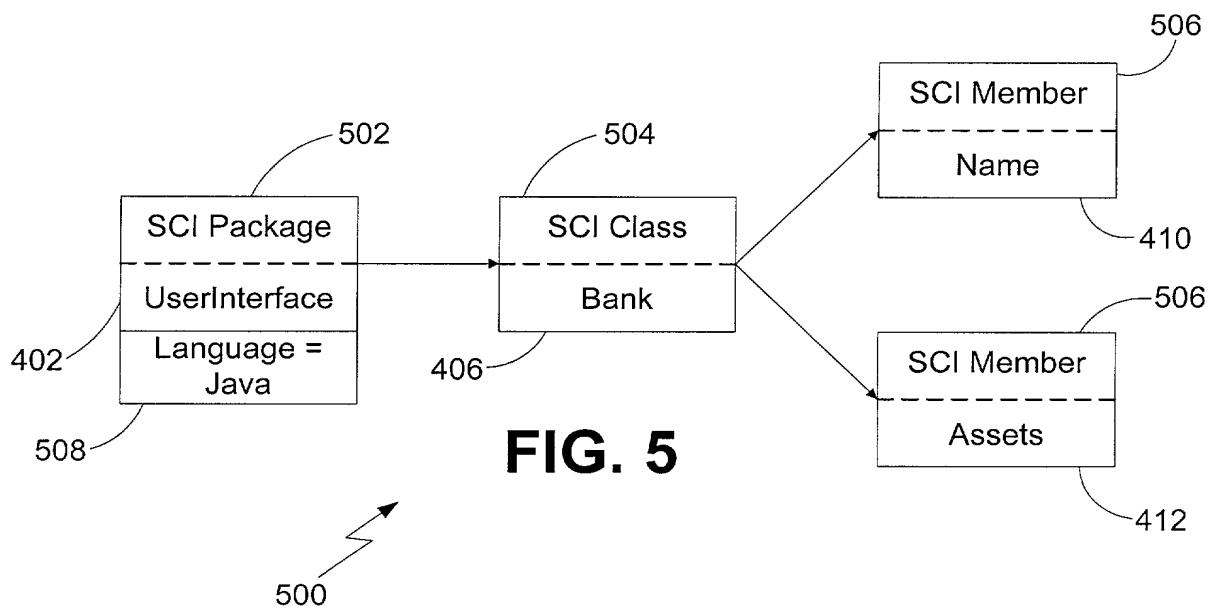
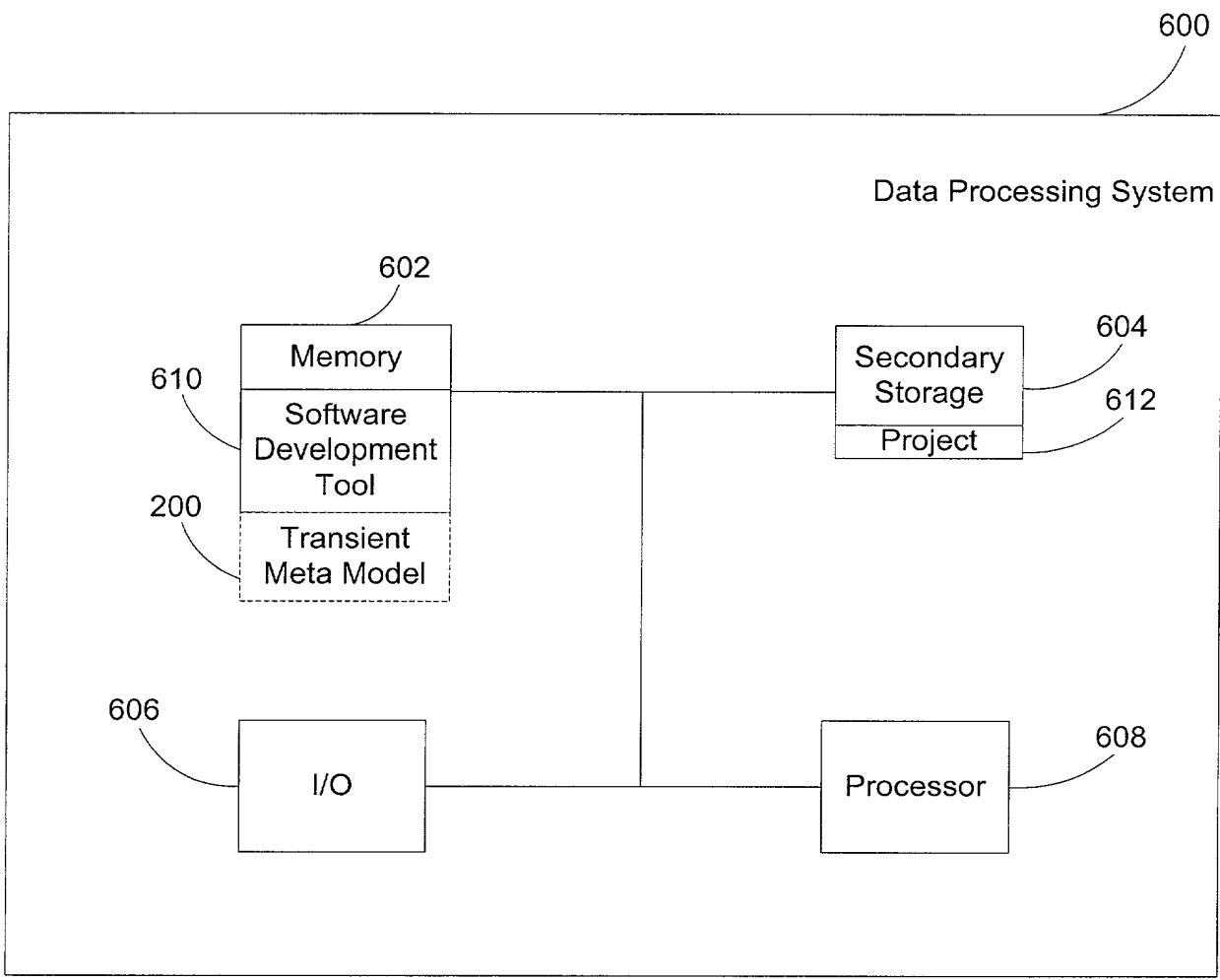


FIG. 5



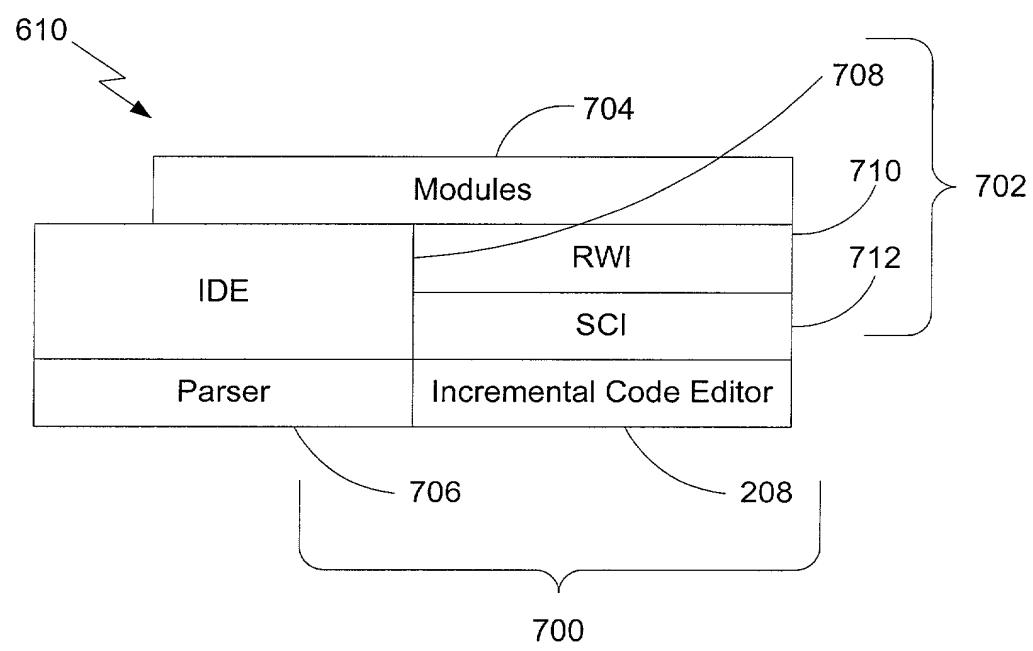


FIG. 7

QA Audit

Title	Abbreviation	Chosen	Severity:
<input type="checkbox"/> Coding Style		<input checked="" type="checkbox"/>	
Access Of Static Members Through Objects	AOSMTO	<input checked="" type="checkbox"/>	800
Assignment To Formal Parameters	ATFP	<input checked="" type="checkbox"/>	
Complex Assignment	CA	<input checked="" type="checkbox"/>	802
Don't Use the Negation Operator Frequently	DUNOF	<input checked="" type="checkbox"/>	
Operator '?' May Not Be Used	OMNBU	<input checked="" type="checkbox"/>	
Provide Incremental In For-Statement or use w...	PIFS	<input checked="" type="checkbox"/>	
Replacement For Demand Imports	RFDI	<input checked="" type="checkbox"/>	
Use Abbreviated Assignment Operator	UAAO	<input checked="" type="checkbox"/>	
Use 'this' Explicitly To Access Class Members	UTETACM	<input checked="" type="checkbox"/>	
<input type="checkbox"/> Critical Errors		<input checked="" type="checkbox"/>	
Avoid Hiding Inherited Attributes	AHIA	<input checked="" type="checkbox"/>	
Avoid Hiding Inherited Static Methods	AHISM	<input checked="" type="checkbox"/>	
Command Query Separation	CQS	<input checked="" type="checkbox"/>	
Hiding Of Names	HON	<input checked="" type="checkbox"/>	
Inaccessible Constructor Or Method Matches	ICCOMM	<input checked="" type="checkbox"/>	
Multiple Visible Declarations With Same Name	MVDWSN	<input checked="" type="checkbox"/>	
Overriding a Non-Abstract Method With an Ab...	ONAMWAM	<input checked="" type="checkbox"/>	
Overriding a Private Method	OPM	<input checked="" type="checkbox"/>	
<input type="button" value="Select all"/> <input type="button" value="Unselect all"/> <input type="button" value="Set defaults"/> <input type="button" value="Save set As..."/> <input type="button" value="Load set..."/>			

AOSMTO - Access Of Static Members Through Objects

Static members should be referenced through class names rather than through objects.

804

FIG. 8A

806

QA Audit

Title	Abbreviation	Chosen	Severity.	Normal
Complex Assignment	CA	<input checked="" type="checkbox"/>		
Don't Use the Negation Operator Frequently	DUNOF	<input checked="" type="checkbox"/>		
Operator '?:' May Not Be Used	OMNBU	<input checked="" type="checkbox"/>		
Provide Incremental In For-Statement or use w...	PIIFS	<input checked="" type="checkbox"/>		
Replacement For Demand Imports	RFDI	<input checked="" type="checkbox"/>		
Use Abbreviated Assignment Operator	UAAO	<input checked="" type="checkbox"/>		

Select all Unselect all Set defaults Save set As... Load set...

CA - Complex Assignment

808

Checks for the occurrence of multiple assignments and assignments to variables within the same expression. Too complex assignments should be avoided since they decrease program readability

Wrong

```
// compound assignment
i *= j++;
k = j = 10;
l = j += 15;
// nested assignment
i = j++ + 20;
i = (j = 25) + 30;
```

810

Tip: Break statement into several ones.

Start Cancel Help

FIG. 8B

806

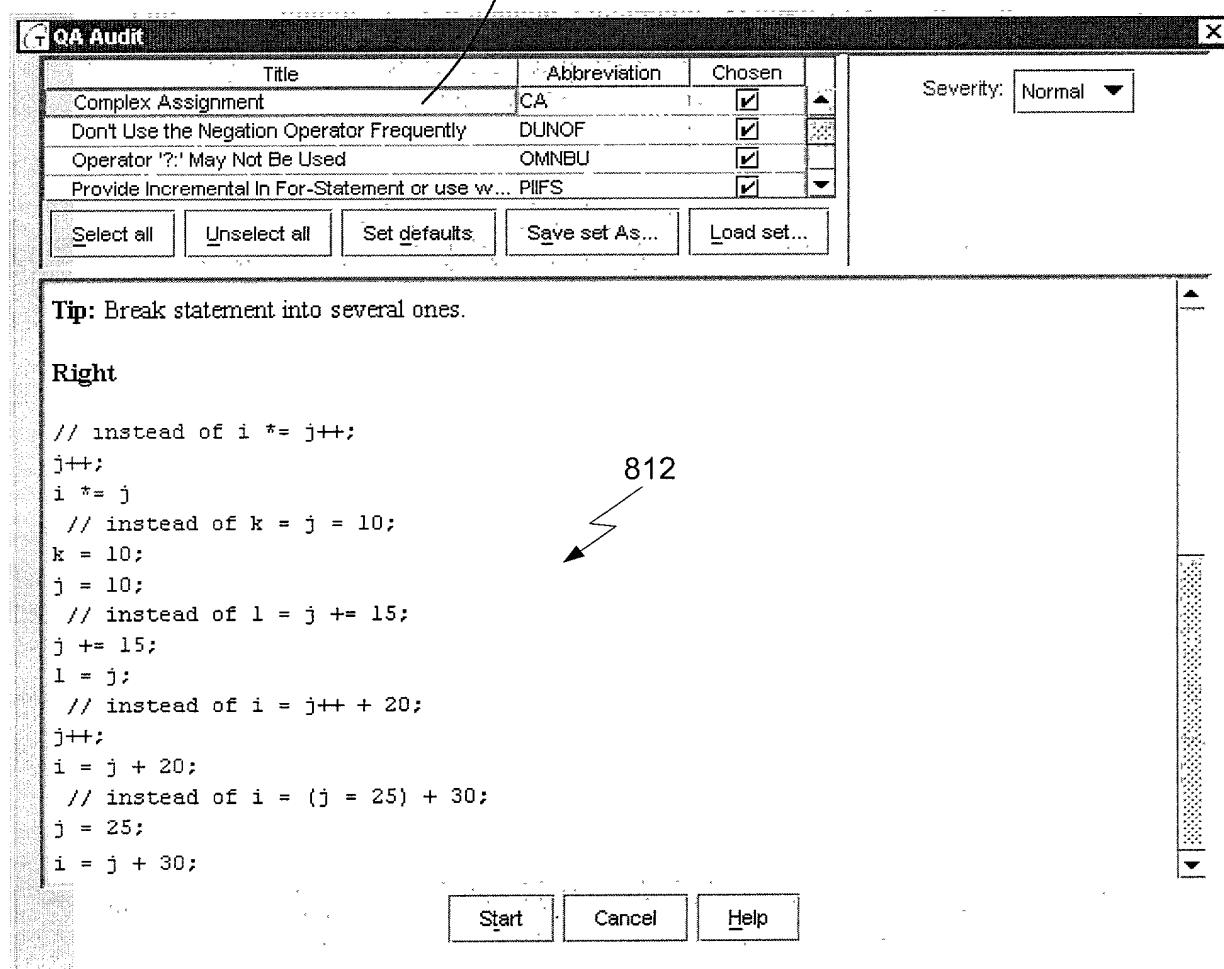


FIG. 8C

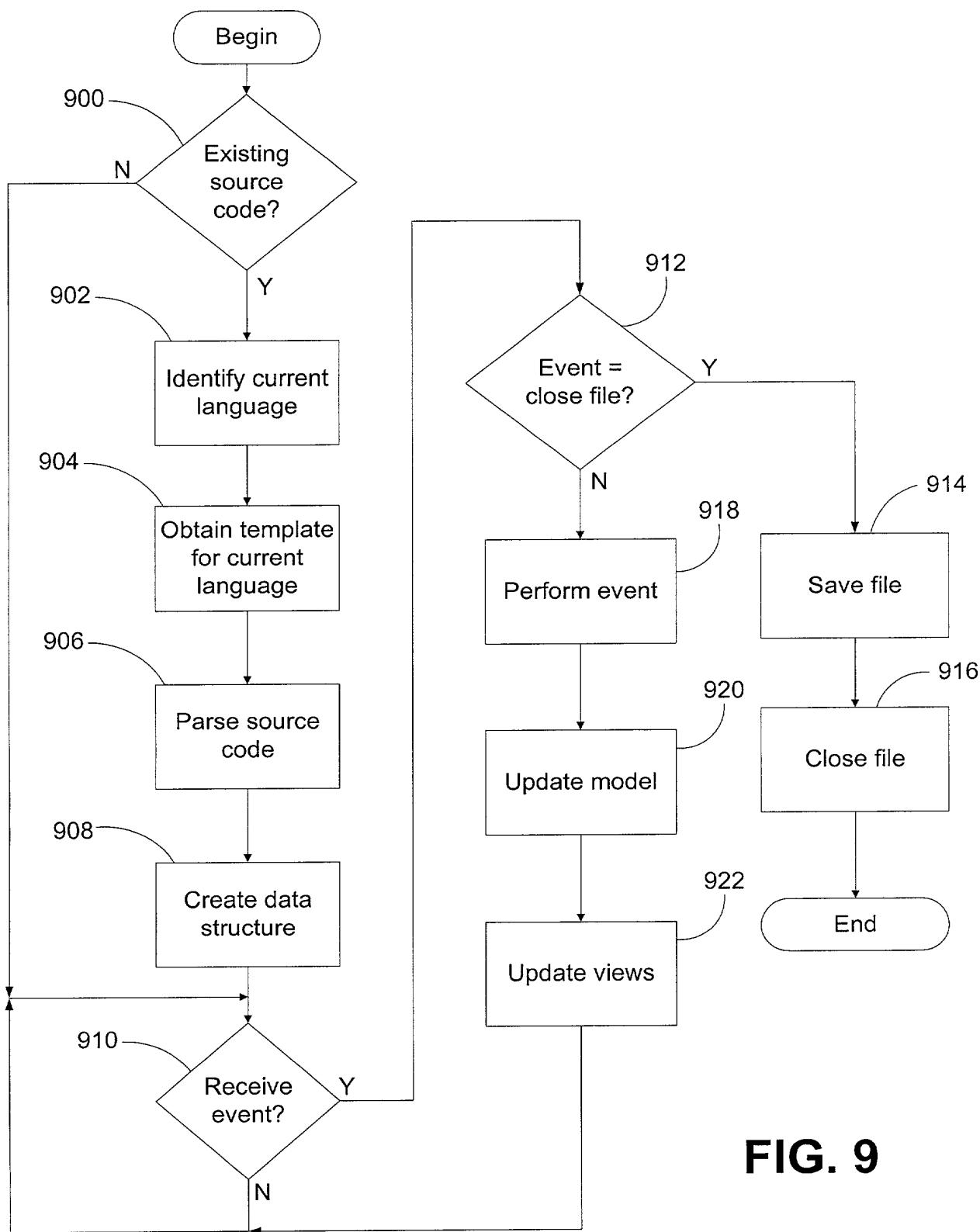


FIG. 9

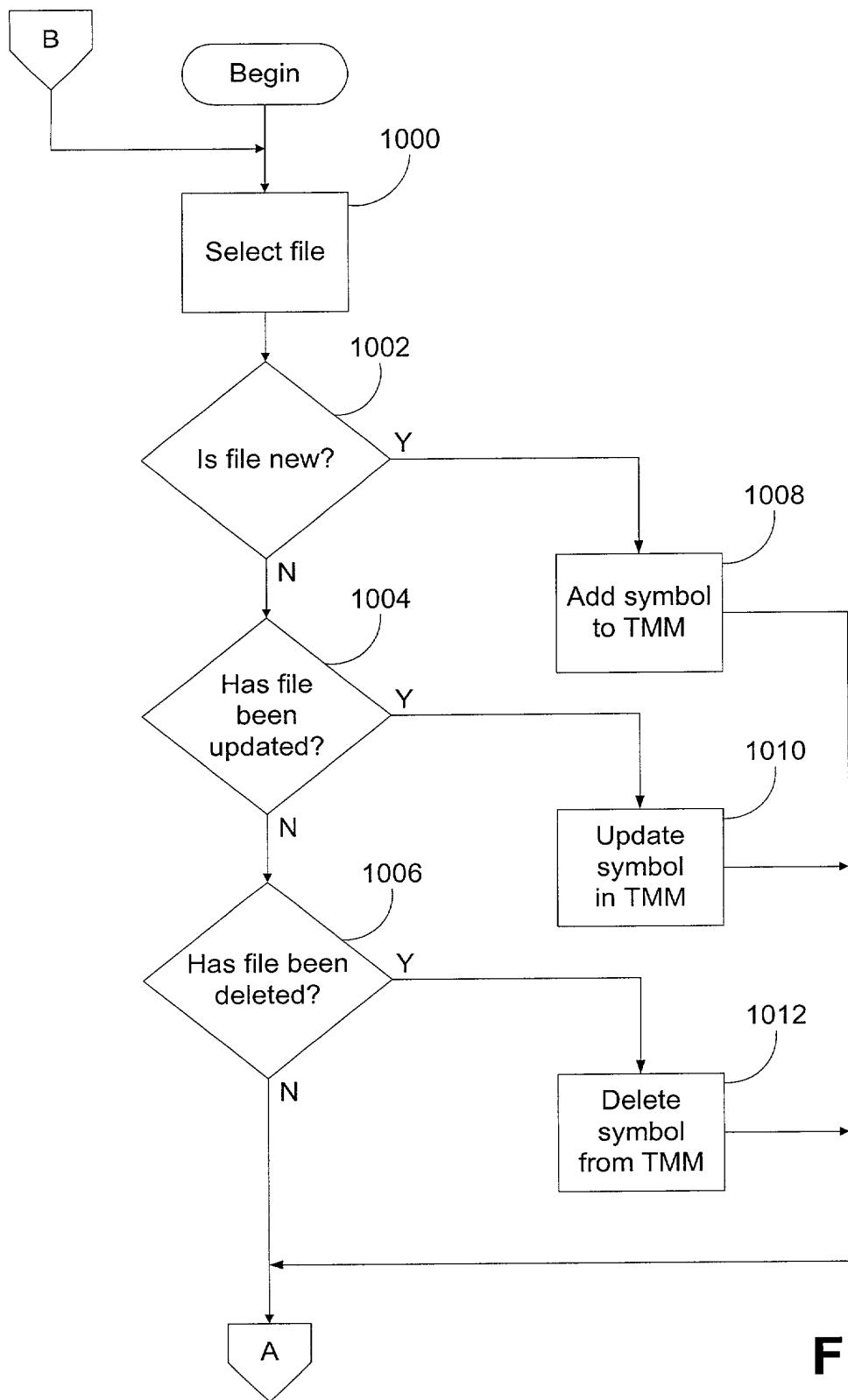


FIG. 10A

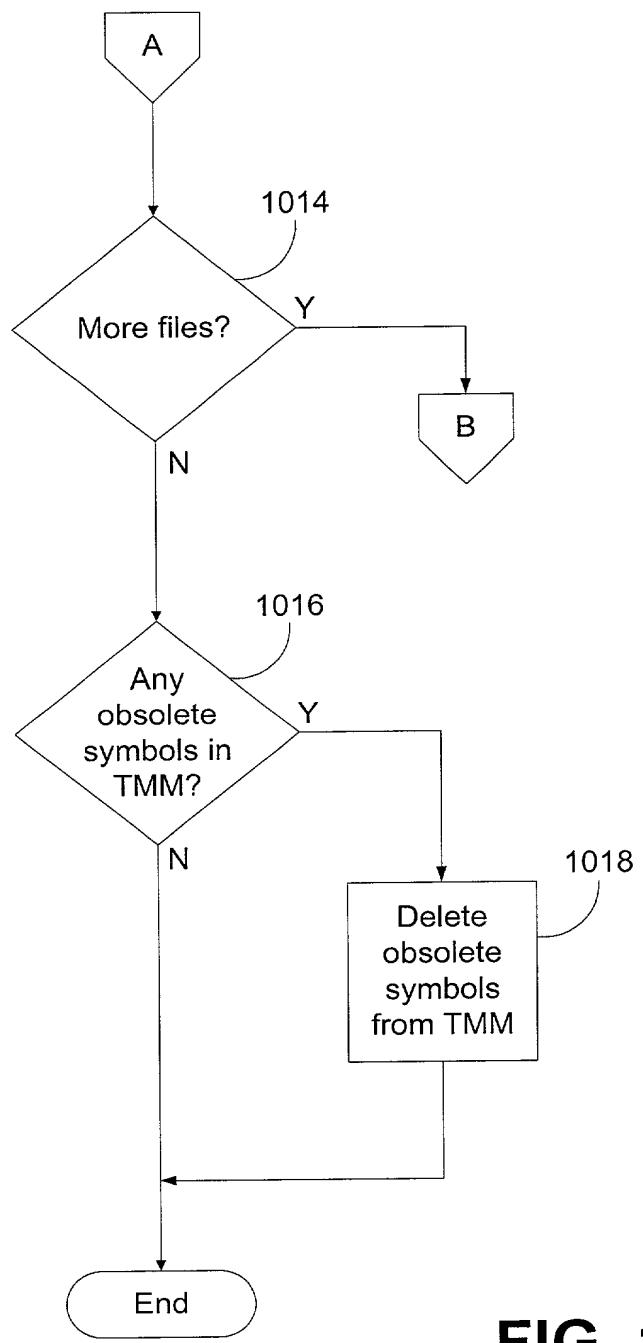


FIG. 10B

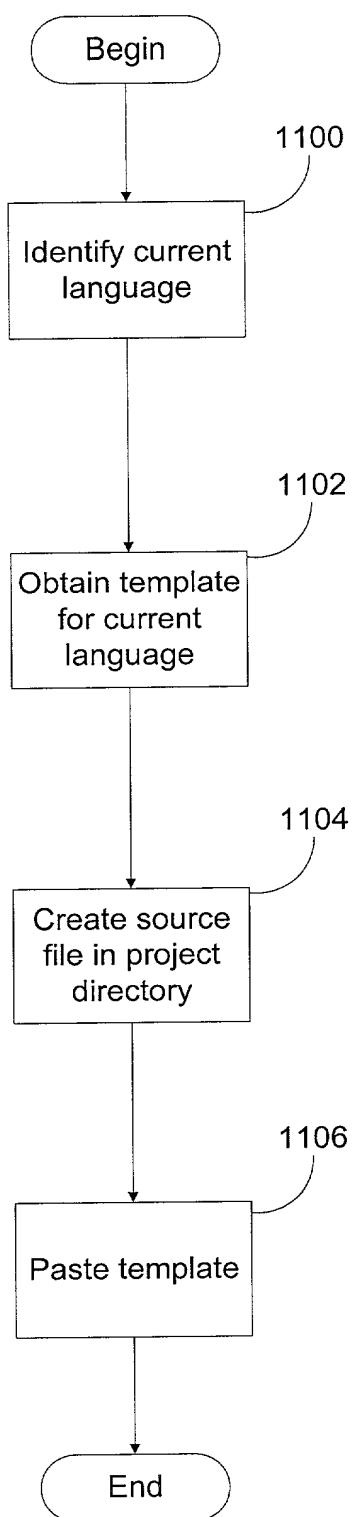


FIG. 11

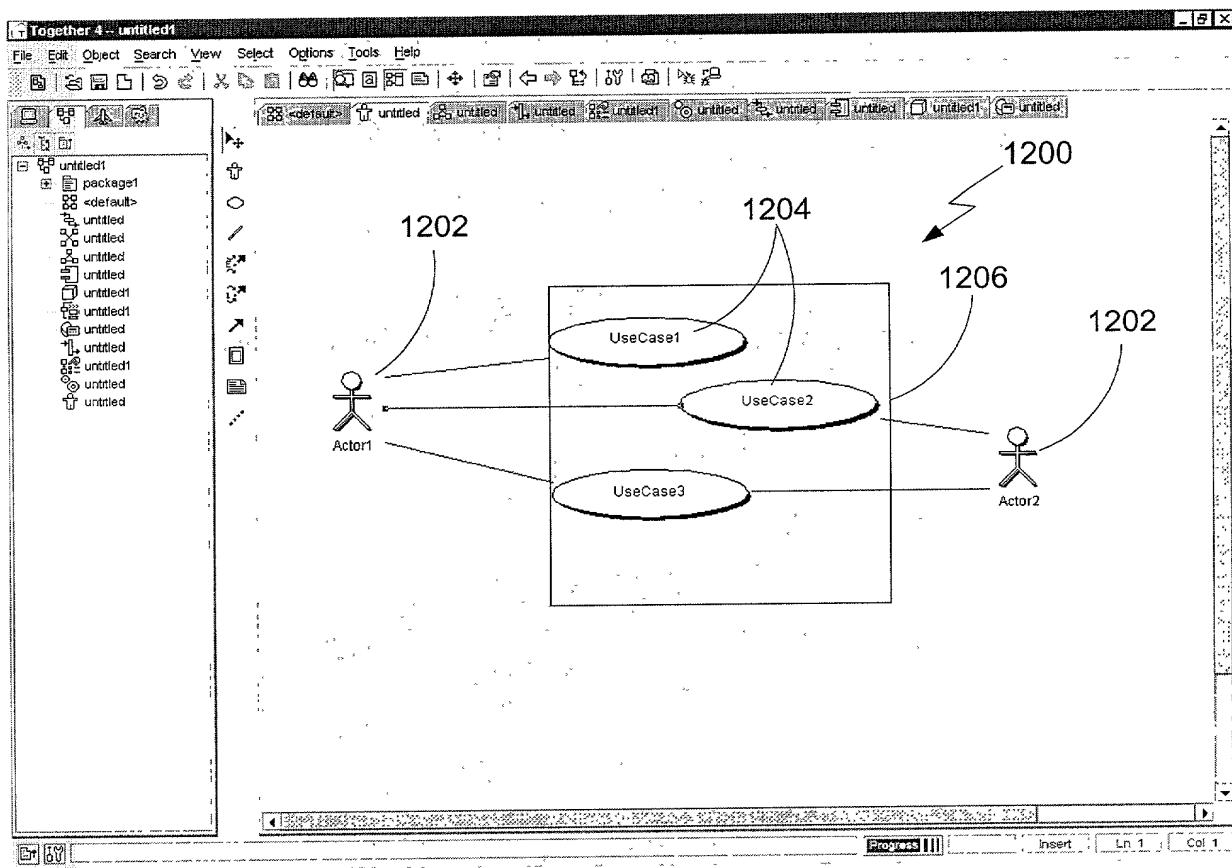


FIG. 12

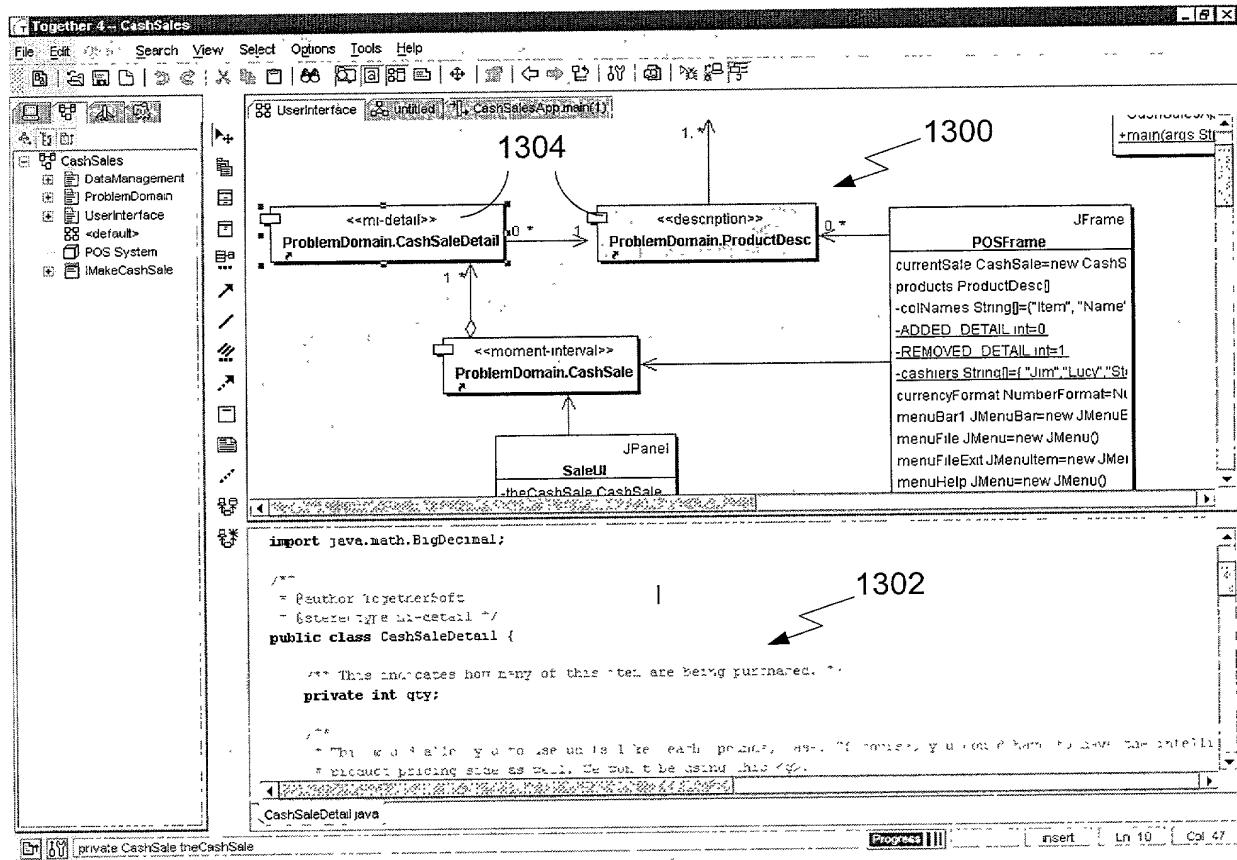


FIG. 13

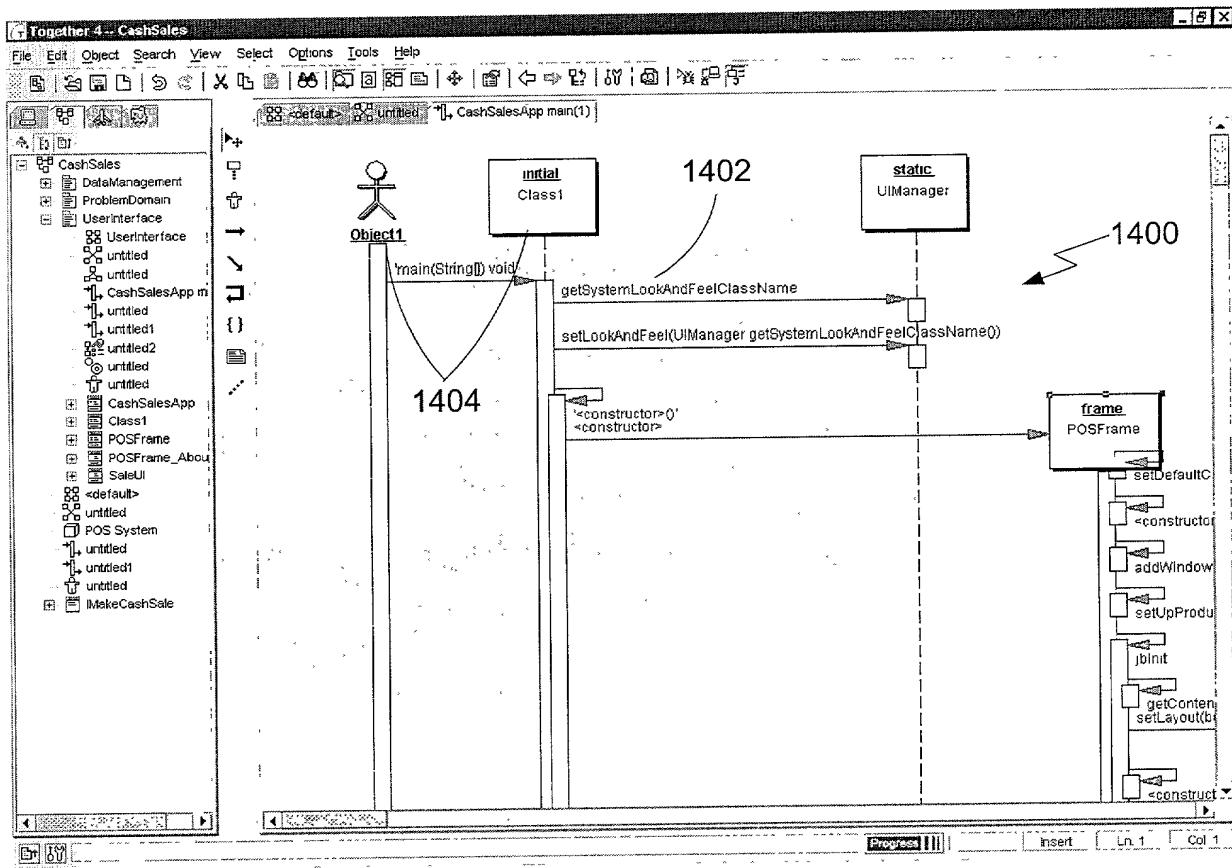


FIG. 14

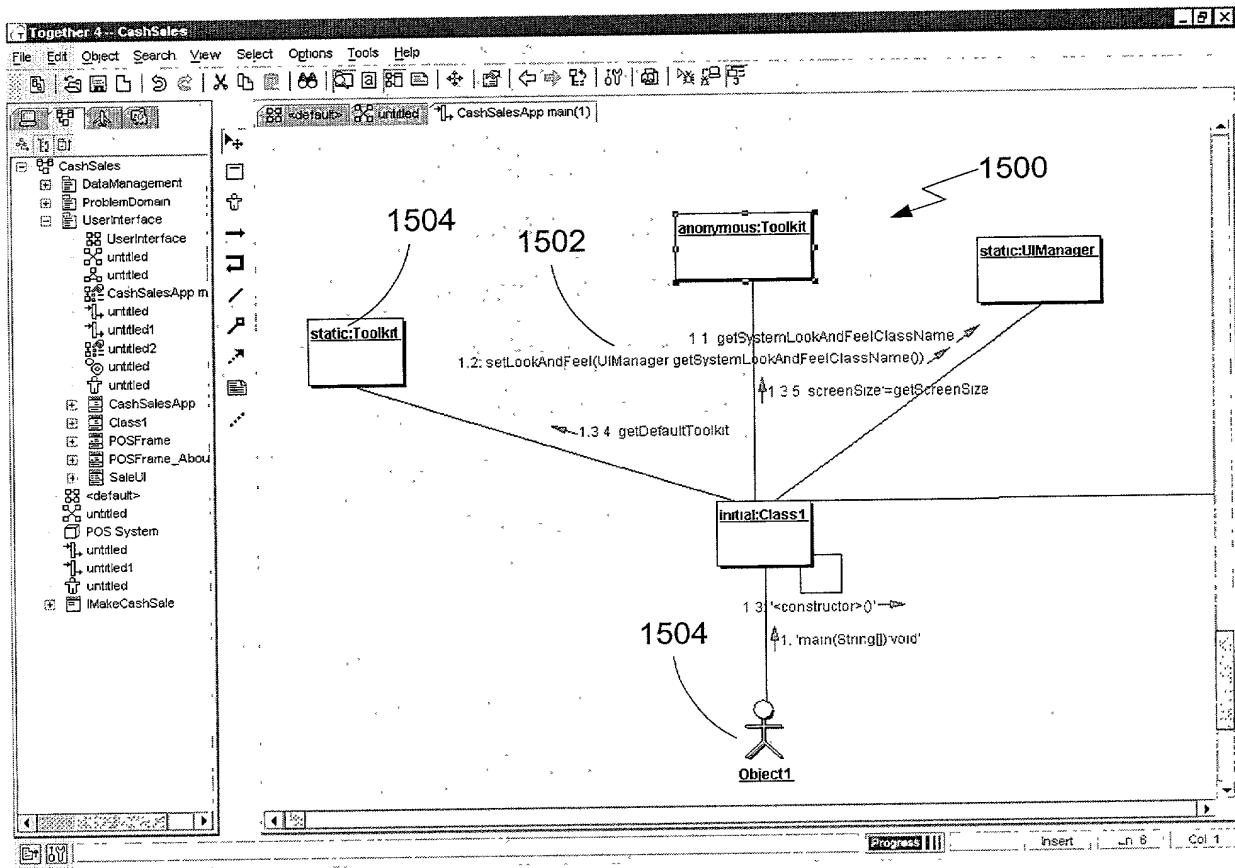


FIG. 15

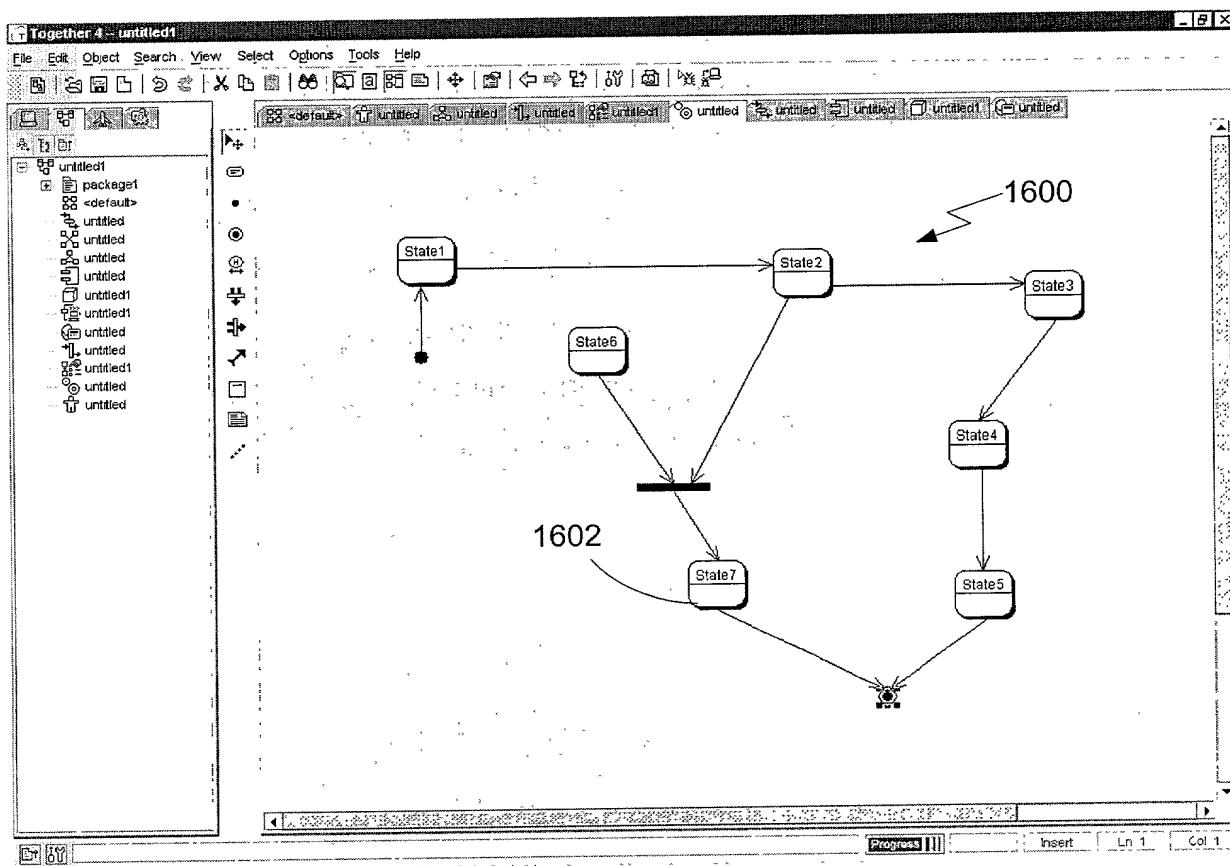


FIG. 16

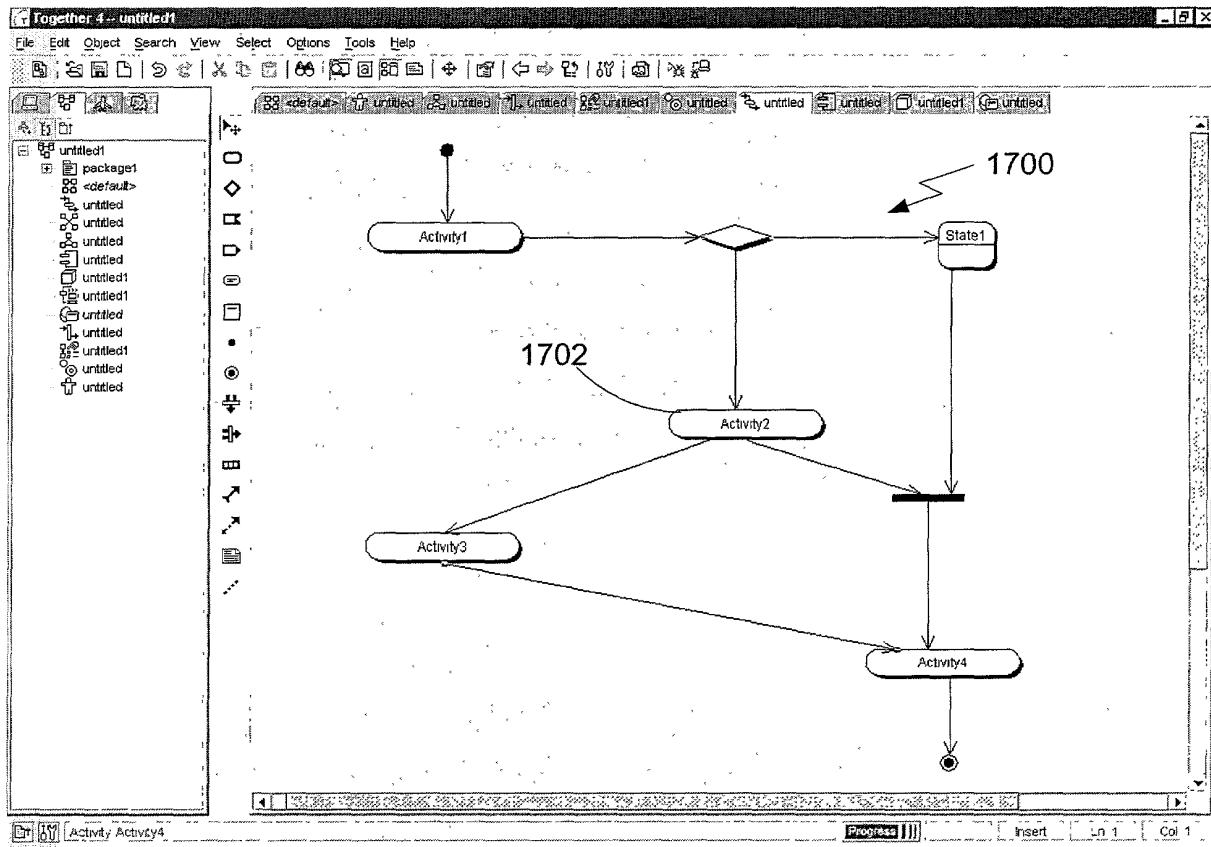


FIG. 17

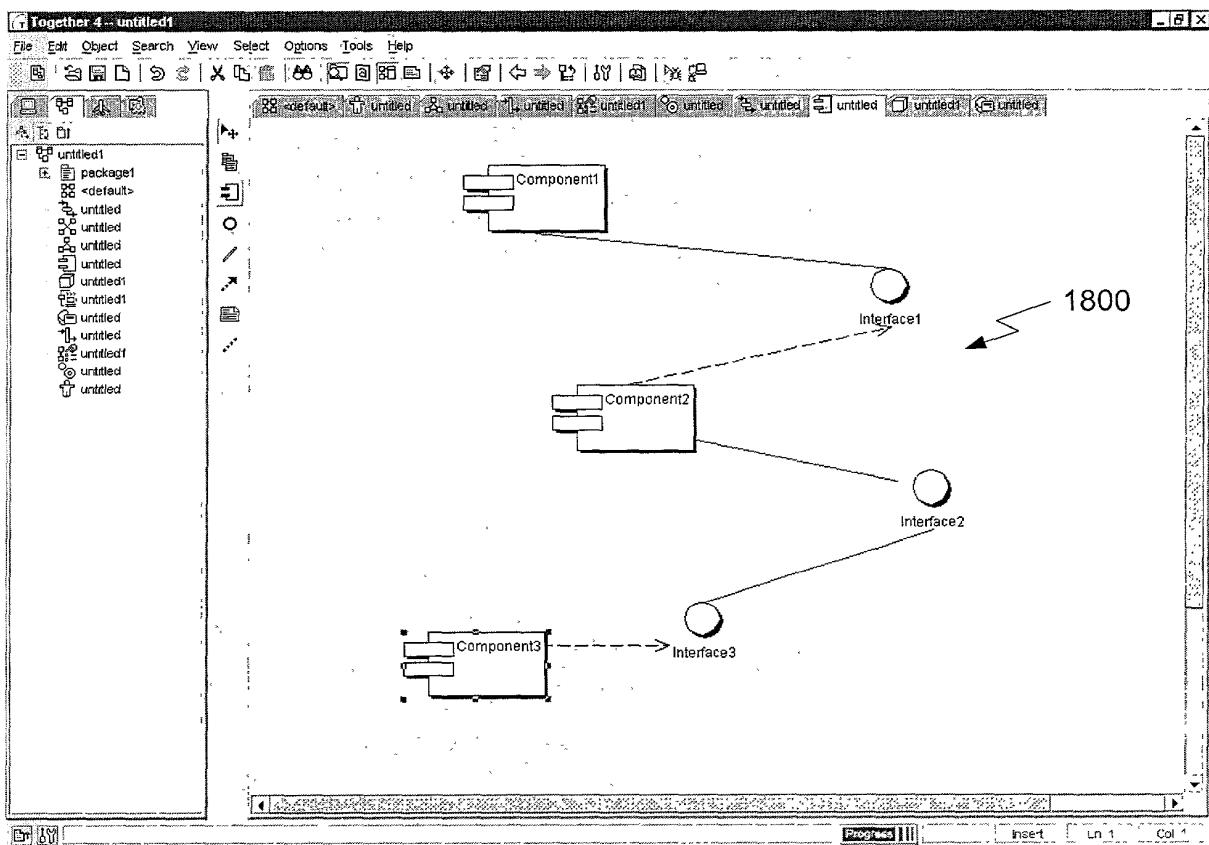


FIG. 18

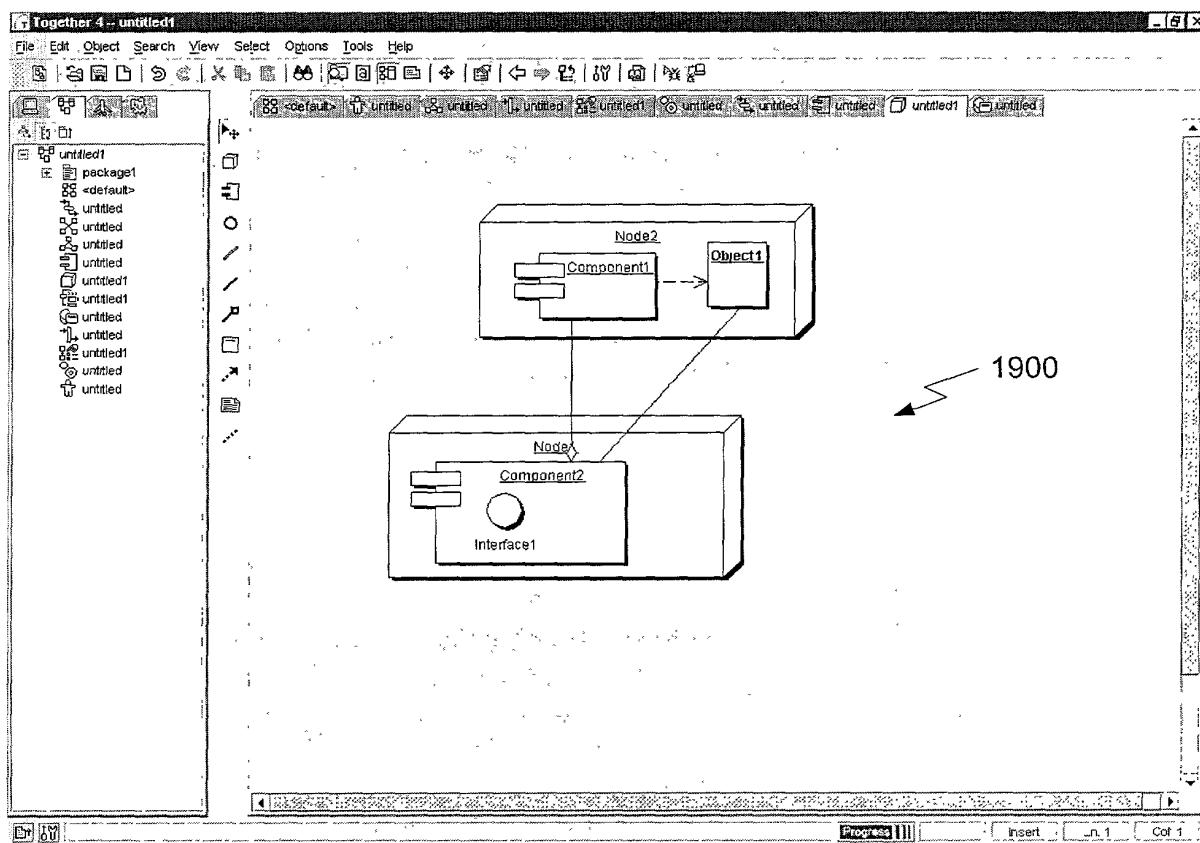


FIG. 19

FIG. 20

```
2002  
public class Sale  
{  
    /**  
     * @link aggregation  
     * @associates <{SaleDetail}>  
     */  
    private Vector InkSaleDetail;  
  
    public void addltem( Product aProd)  
    {  
        2008  
        2004  
        2006  
    }  
  
    public SaleDetail( String barCode)  
    {  
        Product item = Product.lookUp( barCode );  
    }  
    2010  
    2012
```

FIG. 21

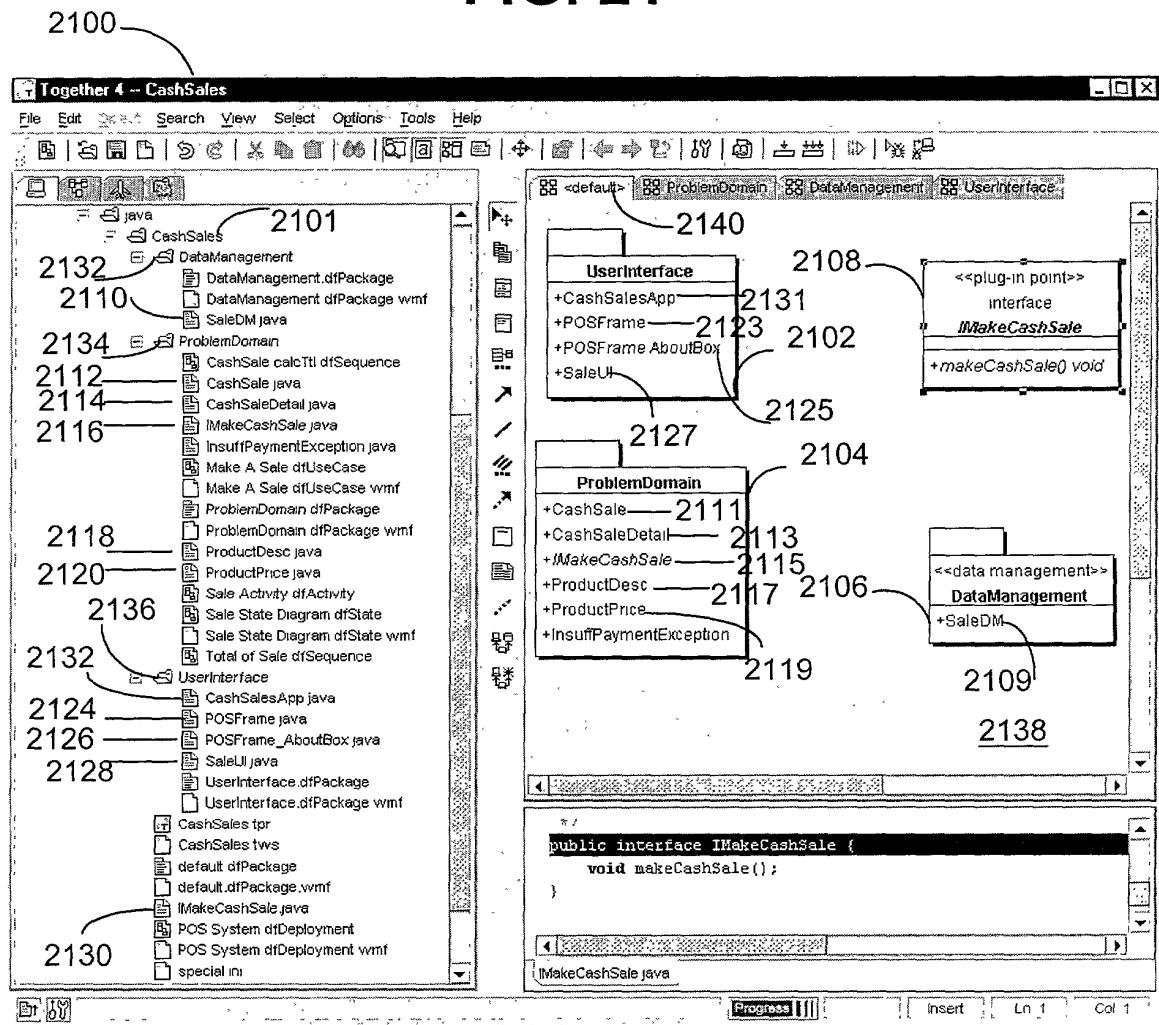


FIG. 22A

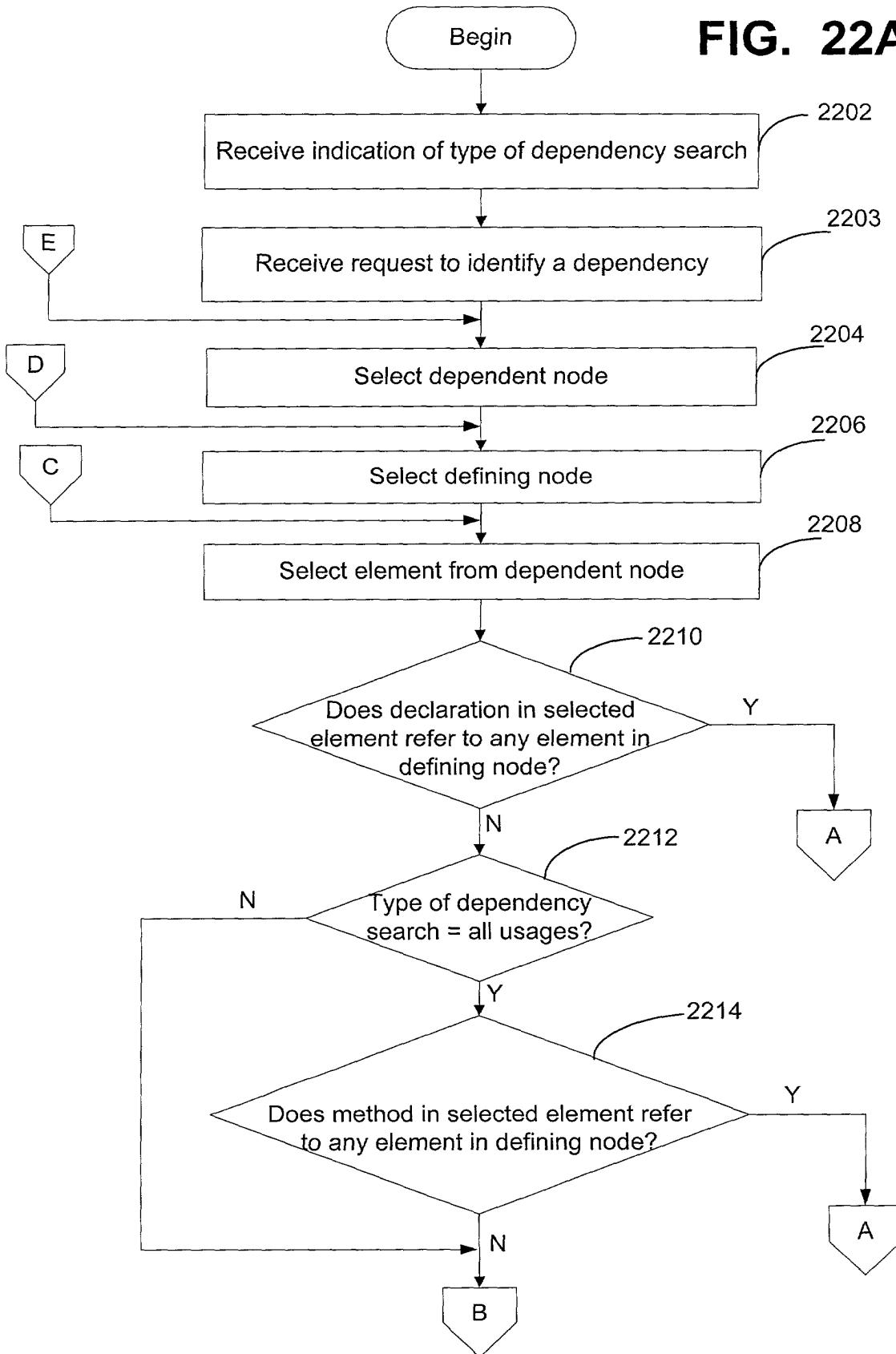


FIG. 22B

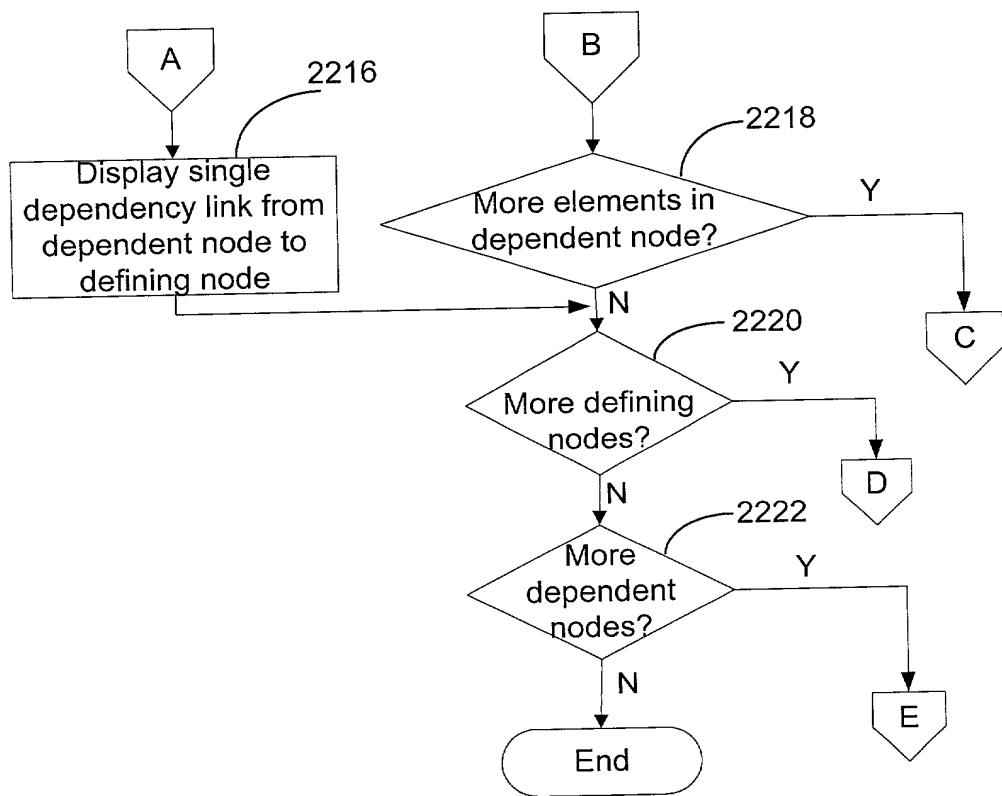


FIG. 23

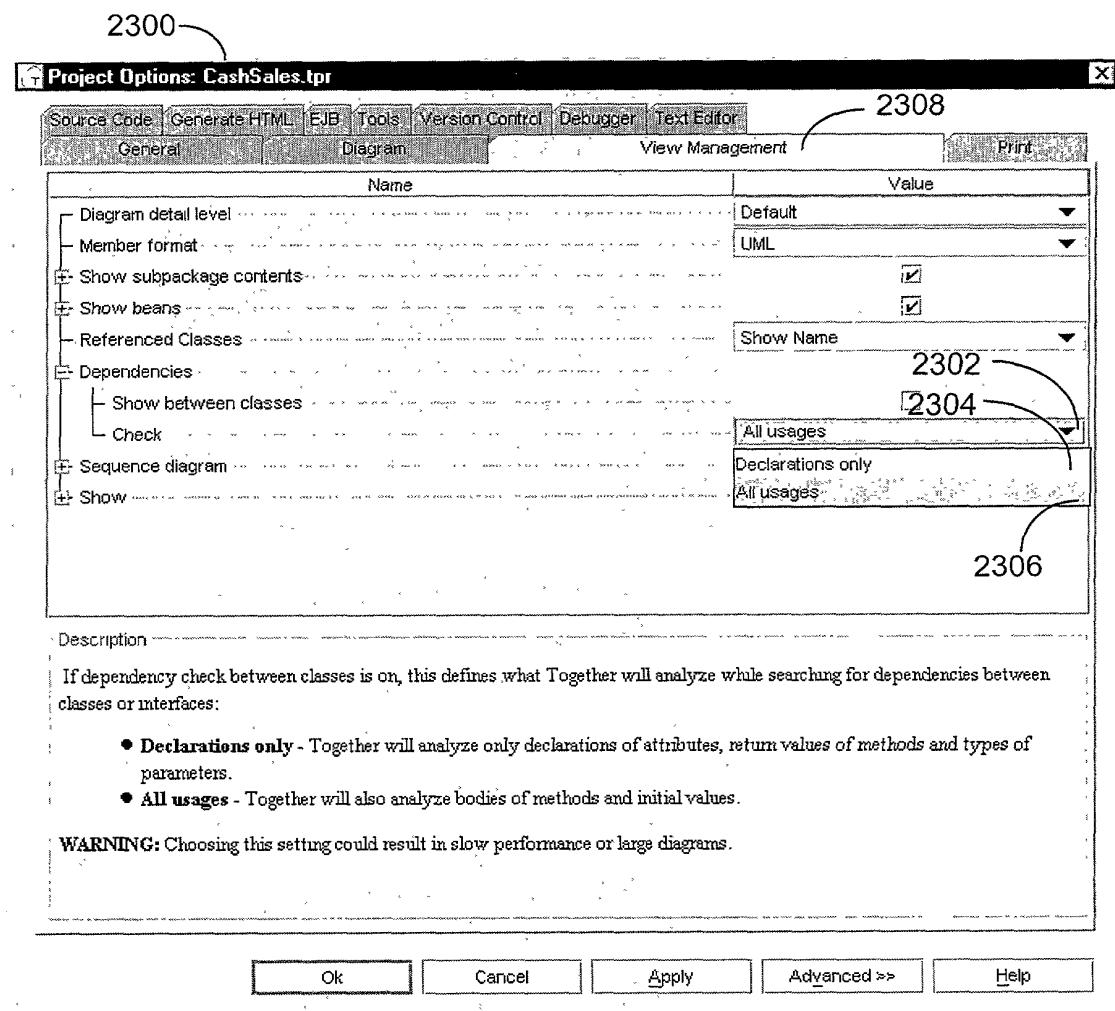


FIG. 24

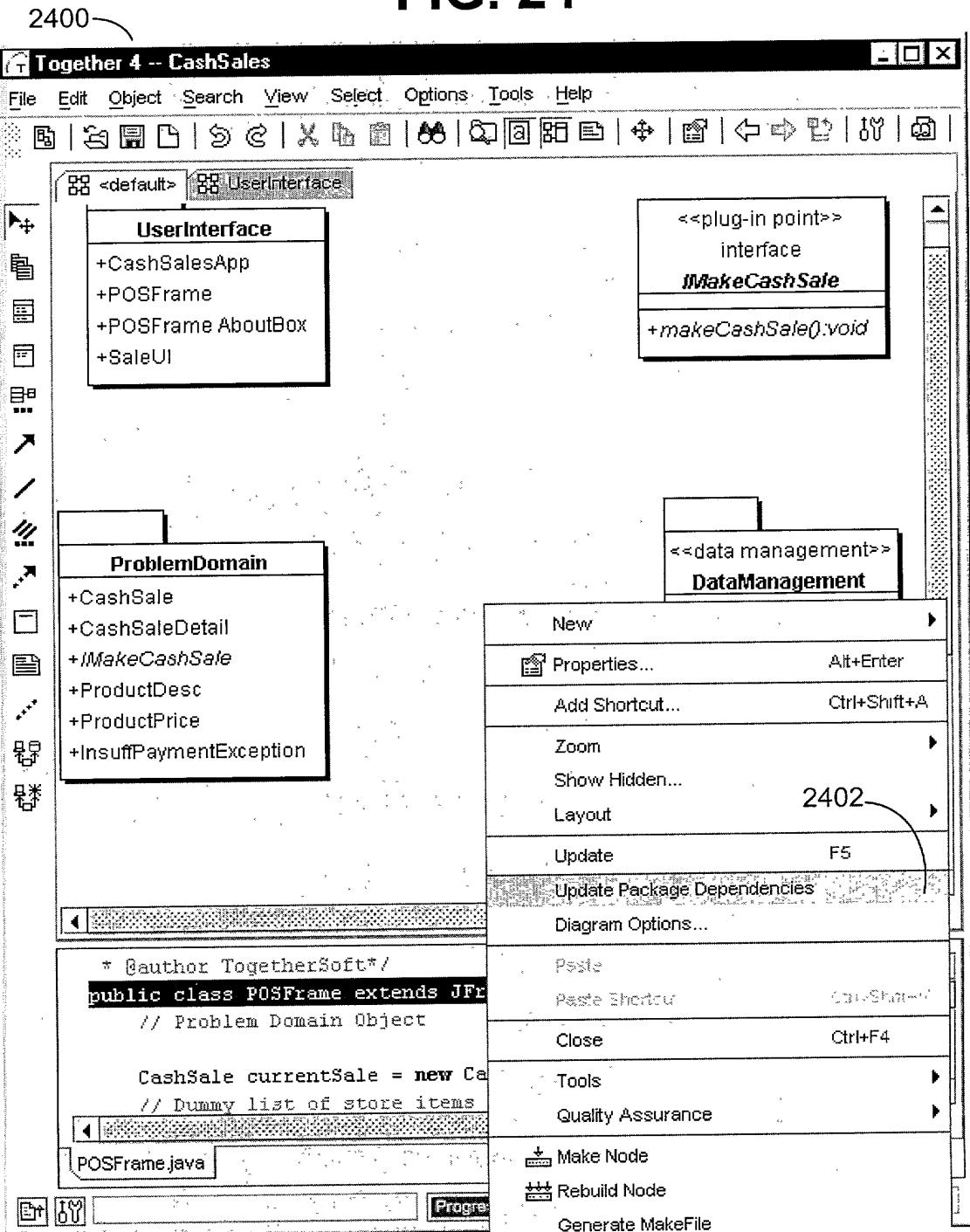


FIG. 25

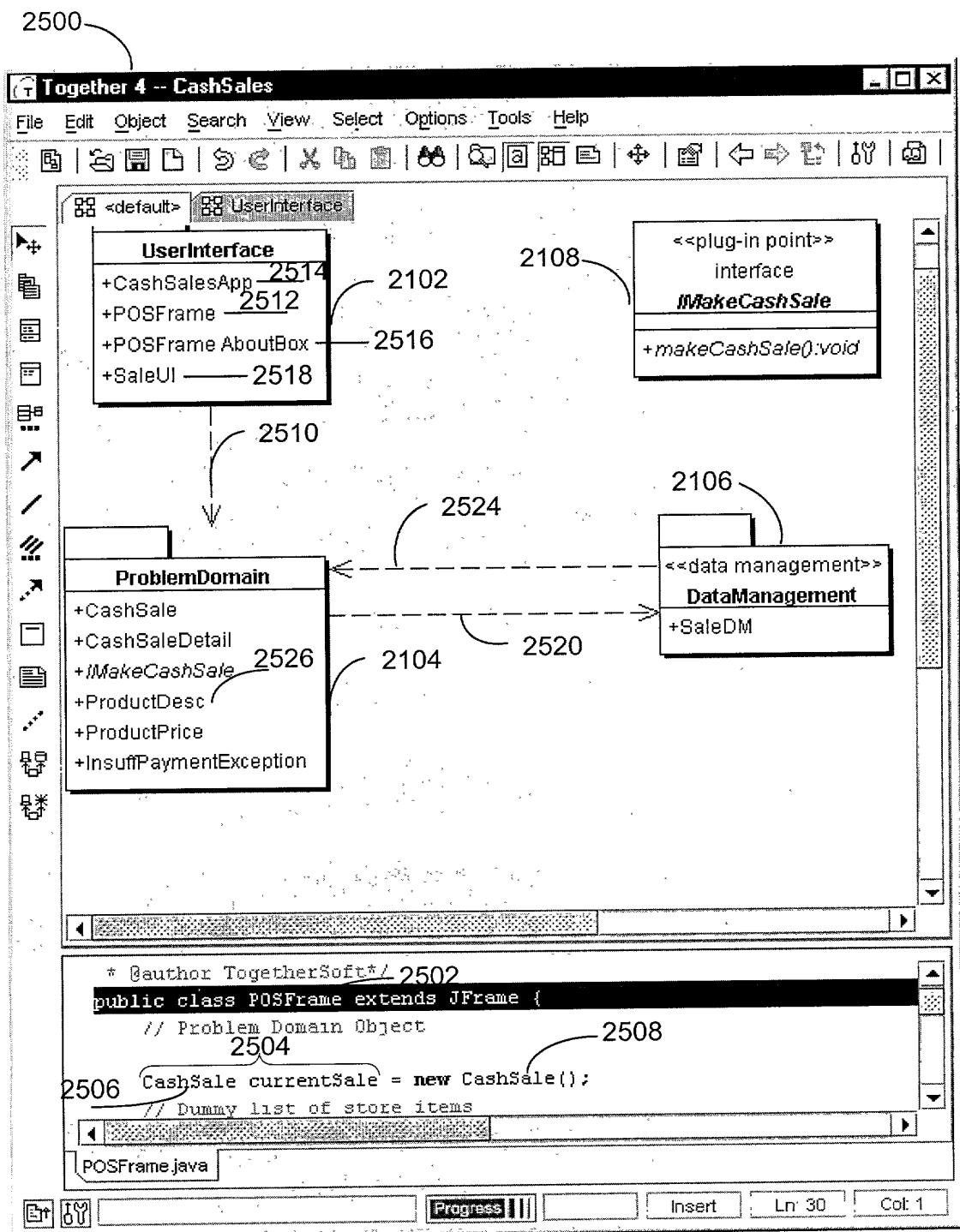


FIG. 26

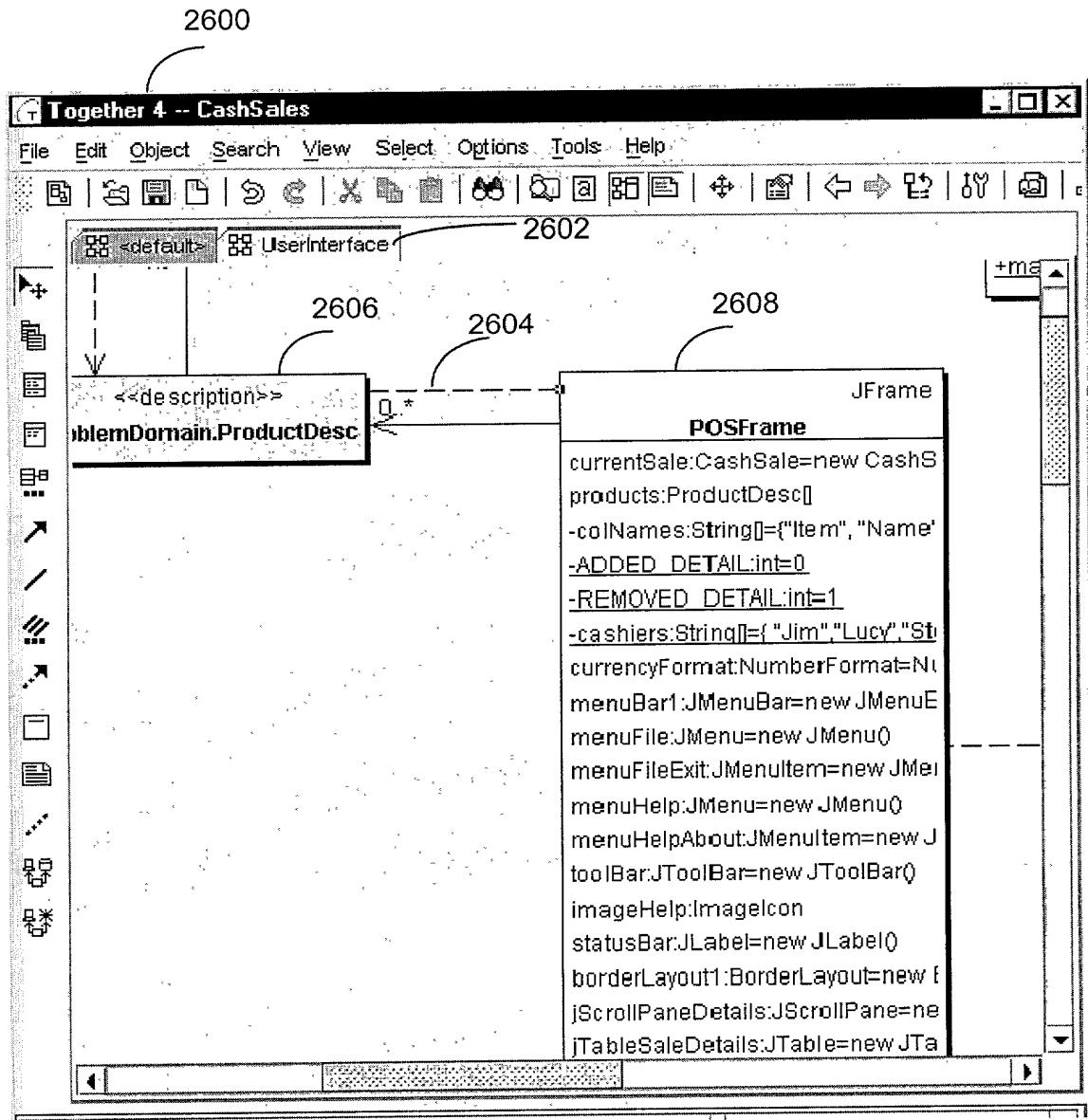


FIG. 27A

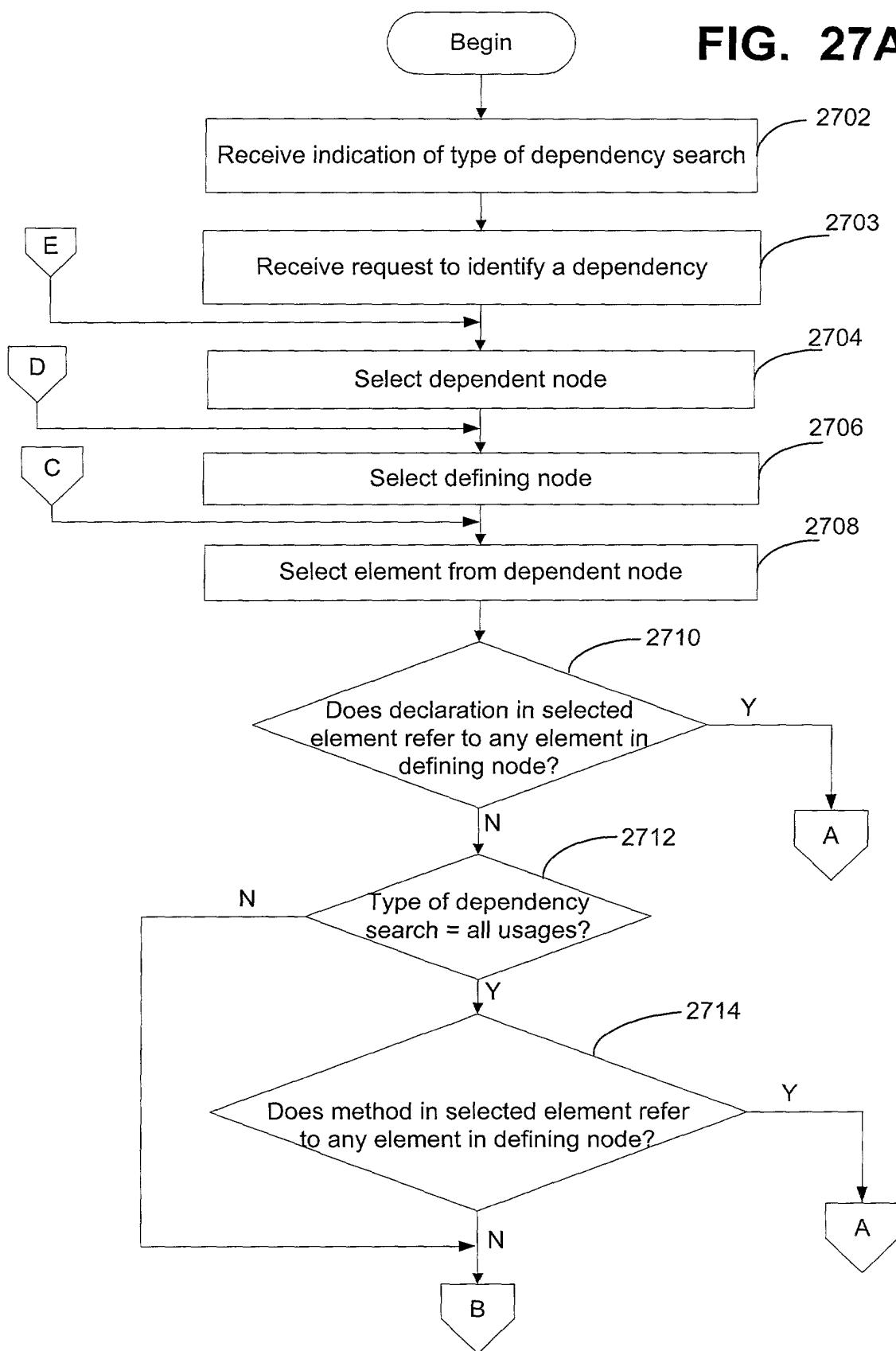


FIG. 27B

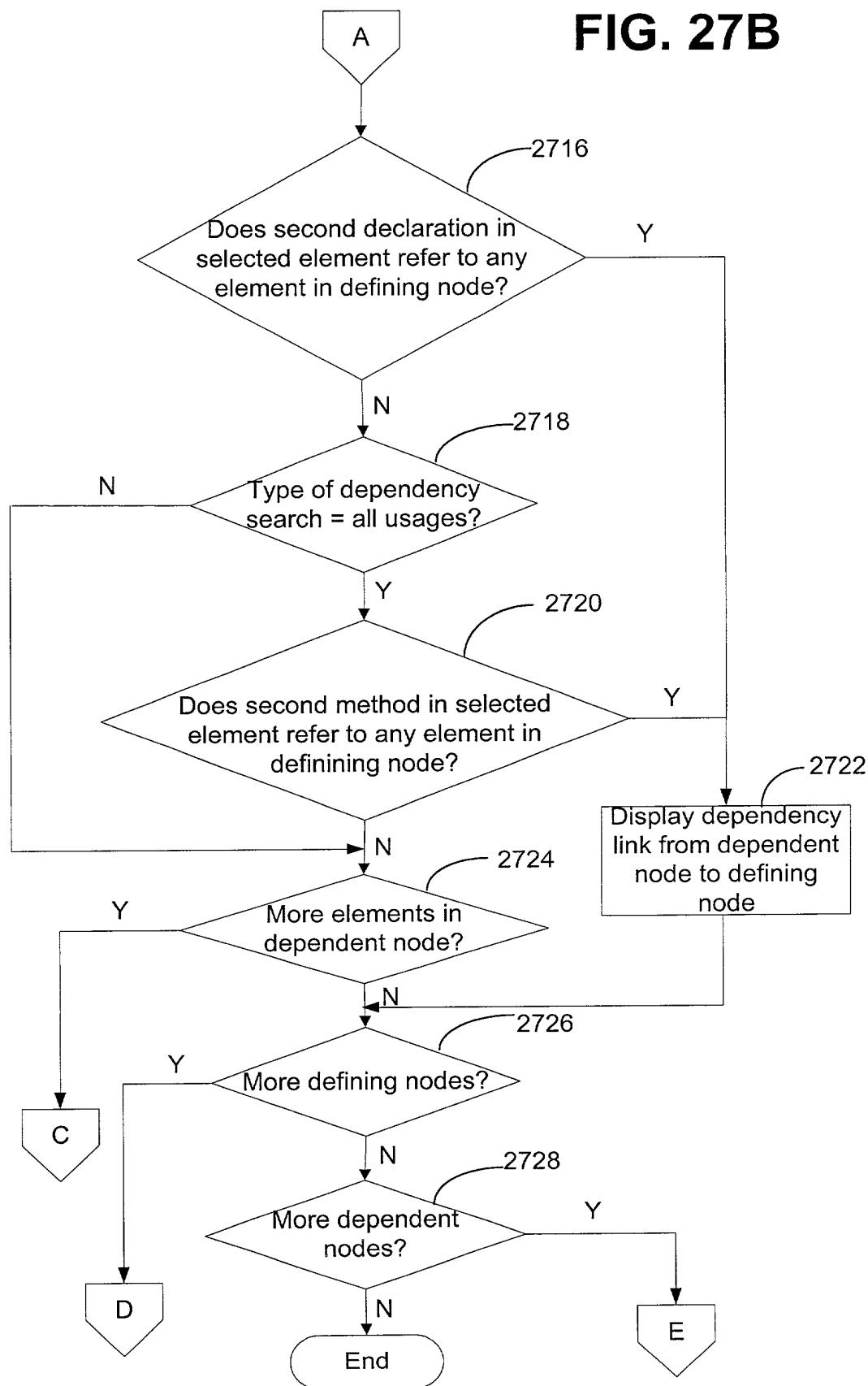


FIG. 28A

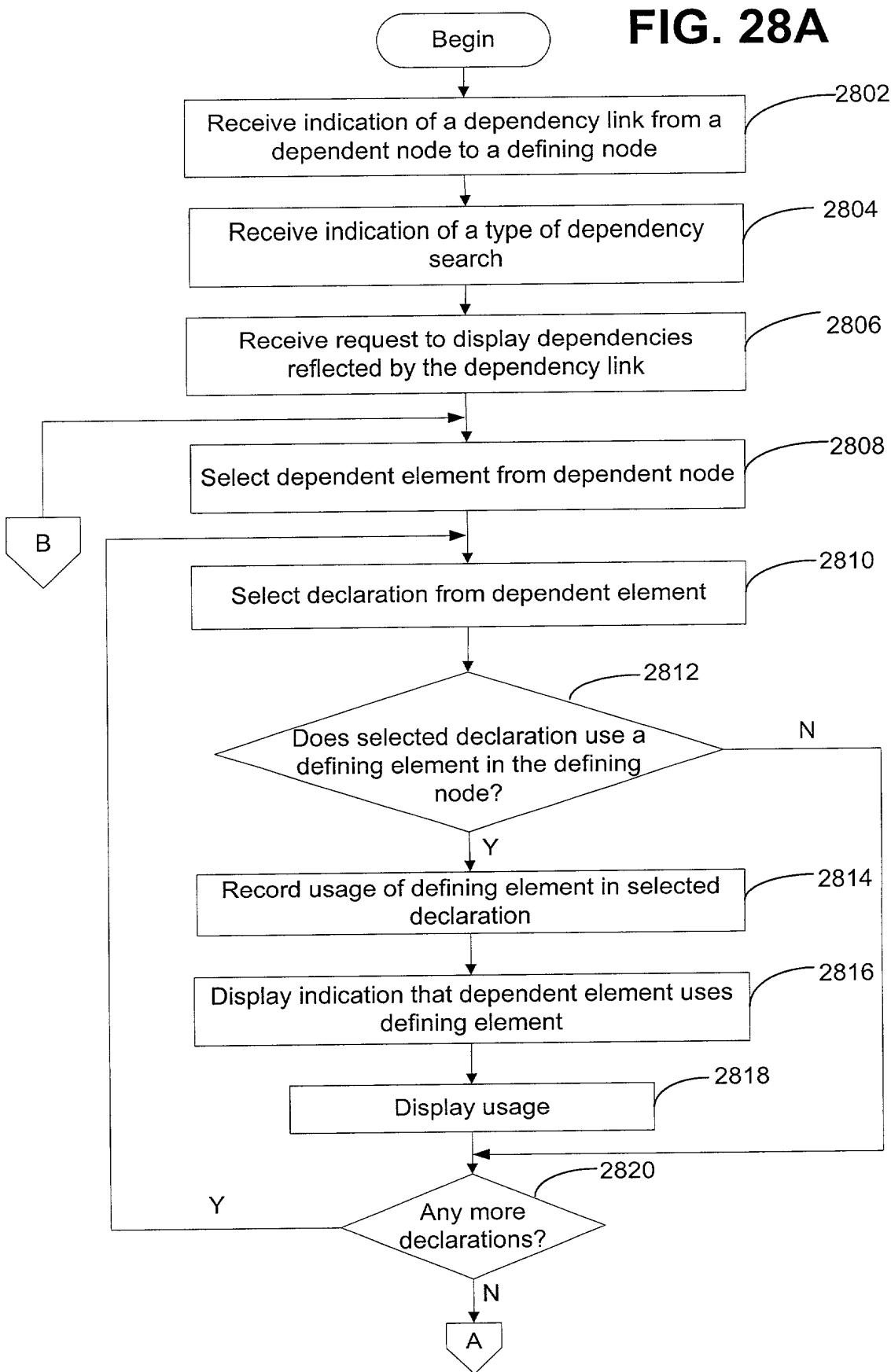


FIG. 28B

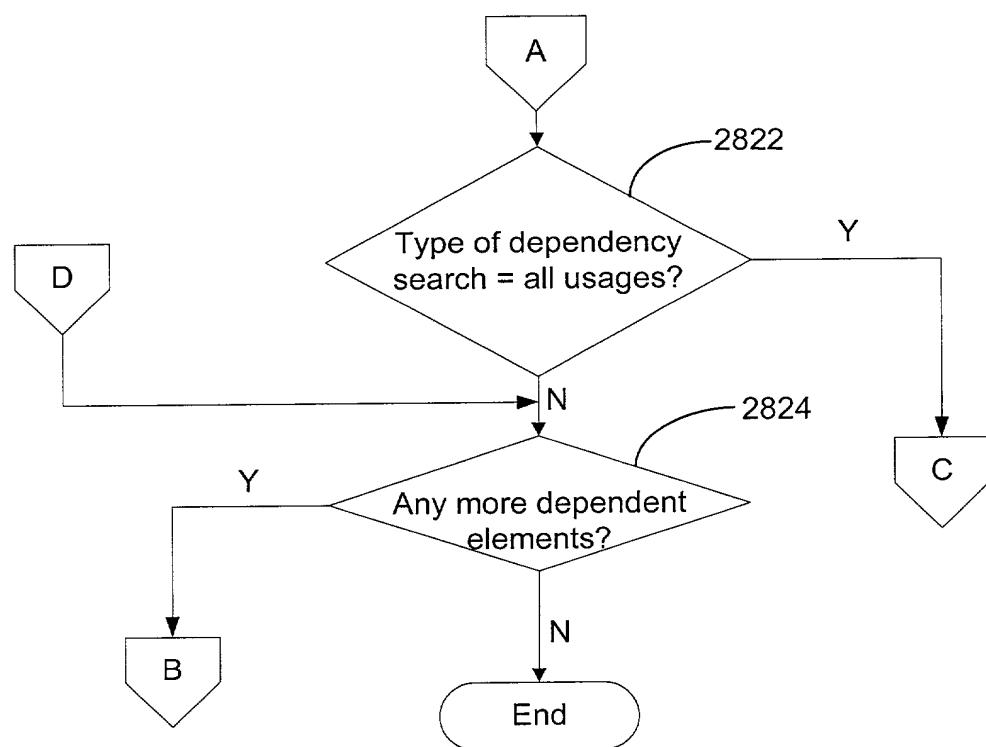


FIG. 28C

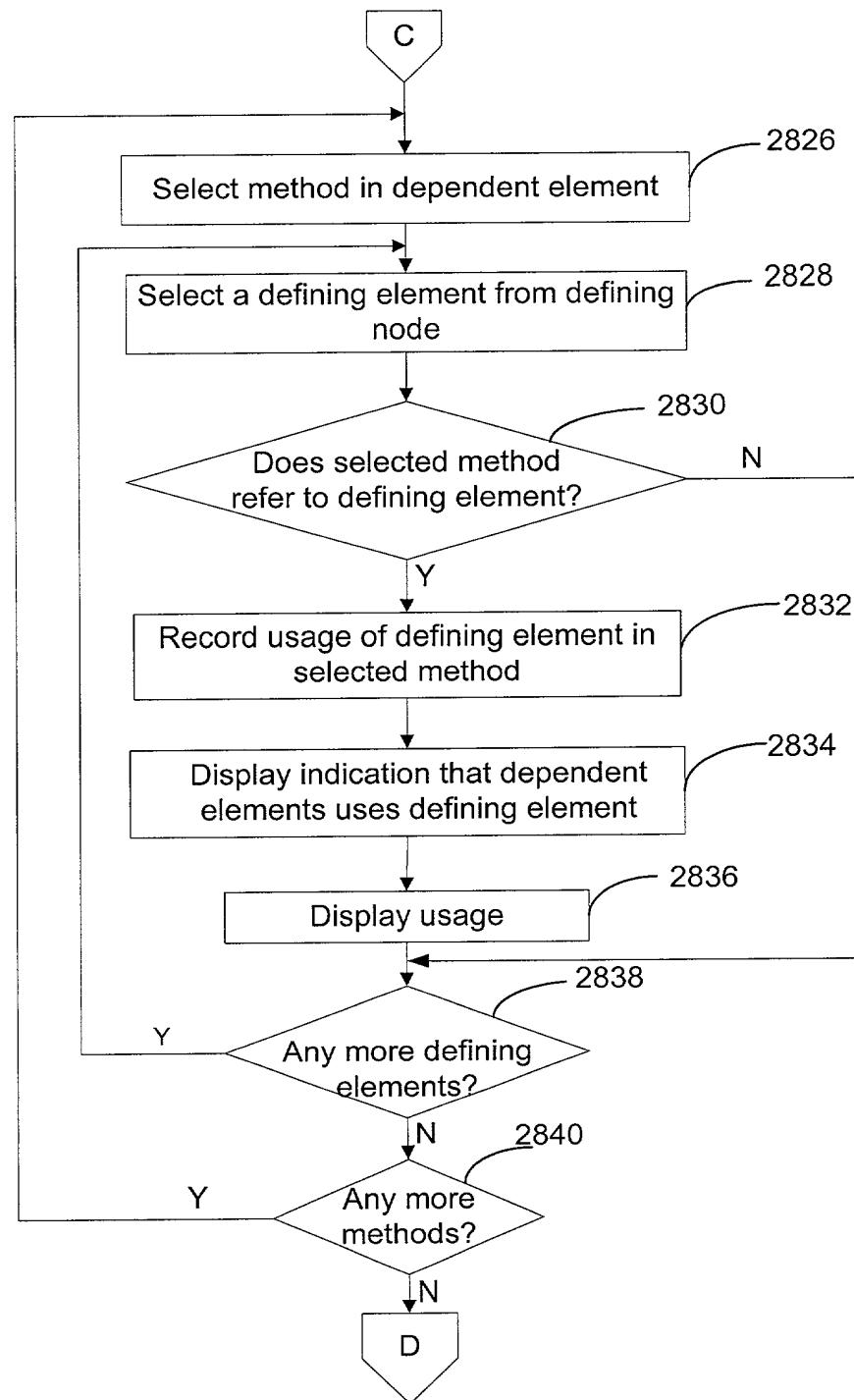


FIG. 29

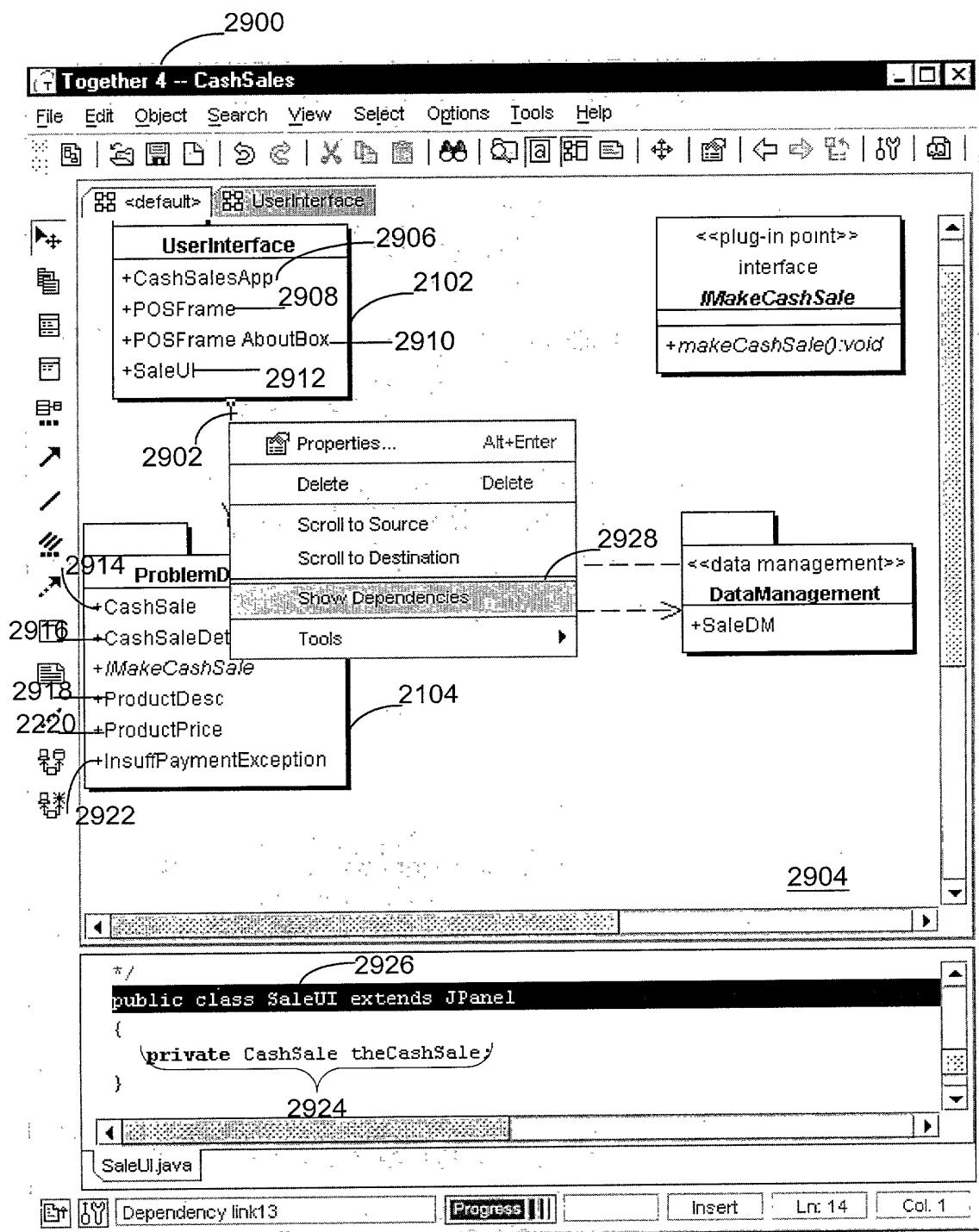


FIG. 30

```

3002
public class POSFrame extends JFrame {
    // Problem Domain Object

    CashSale currentSale = new CashSale();
    // Dummy list of store items

    ProductDesc[] products;
    // Sale Detail Table Column Header
3006
    private final String[] colNames = {"Item", "Name", "Unit", "Qty", "Price"};
    private final static int ADDED_DETAIL = 0;
    private final static int REMOVED_DETAIL = 1;
    // Dummy list of cashiers

    private final static String[] cashiers =
    {
        "Jim", "Lucy", "Steve", "Sarah", "Jon", "Buddy", "Bettie", "Sue", "John", "Ted";
        NumberFormat currencyFormat = NumberFormat.getInstance();
    }

    3010
    private void setUpProducts() {
        products = new ProductDesc[10];
        products[0] = new ProductDesc("1", "Pepsi 24-pack", "Pepsi 24", new BigDecimal(3.99));
        products[1] = new ProductDesc("2", "Lays Ridges", "Lays", new BigDecimal(1.99));
        products[2] = new ProductDesc("3", "Vienna Sausages", "Vienna Sausages",
            new BigDecimal(2.99));
        products[3] = new ProductDesc("4", "White Popcorn", "White Popcorn",
            new BigDecimal(1.30));
        products[4] = new ProductDesc("5", "Soy Burgers", "Soy Burger", new BigDecimal(5.99));
        products[5] = new ProductDesc("6", "Cat Chow", "Cat Chow", new BigDecimal(9.99));
        products[6] = new ProductDesc("7", "Puppy Chow", "Puppy Chow", new BigDecimal(12.99));
        products[7] = new ProductDesc("8", "Finch Food", "Finch Food", new BigDecimal(1.59));
        products[8] = new ProductDesc("9", "Rice Krispies", "Rice Krispies", new BigDecimal(3.30));
        products[9] = new ProductDesc("10", "Fruit Loops", "Fruit Loops", new BigDecimal(3.49));
    }
3008
3012
3014
3016

```

The diagram illustrates the flow of the code with various annotations:

- Line 3002:** An annotation above the class definition.
- Line 3004:** An annotation above the declaration of the `products` array.
- Line 3006:** An annotation above the first few lines of the class body.
- Line 3010:** An annotation above the start of the `setUpProducts()` method.
- Line 3012:** A brace annotation spanning the entire `setUpProducts()` method body.
- Line 3014:** An annotation above the assignment of `products[0]`.
- Line 3016:** An annotation above the assignment of `products[9]`.

FIG. 31

```
3102
public class ProductDesc {  
  
    /** Use it if you need to identify Products as specific types. */  
    private int type;  
  
    /** Product name. For example: Goetze's Caramel Cremes */  
    private String name;  
  
    /** This is the unique identifying number. Something like a UPC for retail  
     * products. */  
    private String itemNumber;  
  
    /** Default price. */  
    private BigDecimal defaultPrice;  
  
    /** Some prose describing the product in all its glory. */  
    private String description;  
  
    /**  
     * List of prices. If this list has elements, then they are checked. Otherwise,  
     * the default price is used. <p>  
     * @supplierCardinality 1..*  
     * @associates <b>ProductPrice</b>  
     */  
    private Vector priceObjects;  
  
    /* ======  
     * Constructors  
     * ====== */  
  
3104    /** Constructor requires all parameters. Type is defaulted to 0 since we  
        aren't using it. */  
    public ProductDesc(String anItemNum, String aDesc, String aName,  
                      BigDecimal aPrice) {  
        type = 0; // not currently used  
        itemNumber = anItemNum;  
        description = aDesc;  
        name = aName;  
        defaultPrice = aPrice;  
        priceObjects = new Vector();  
    } // END ProductDesc(...)
```

FIG. 32

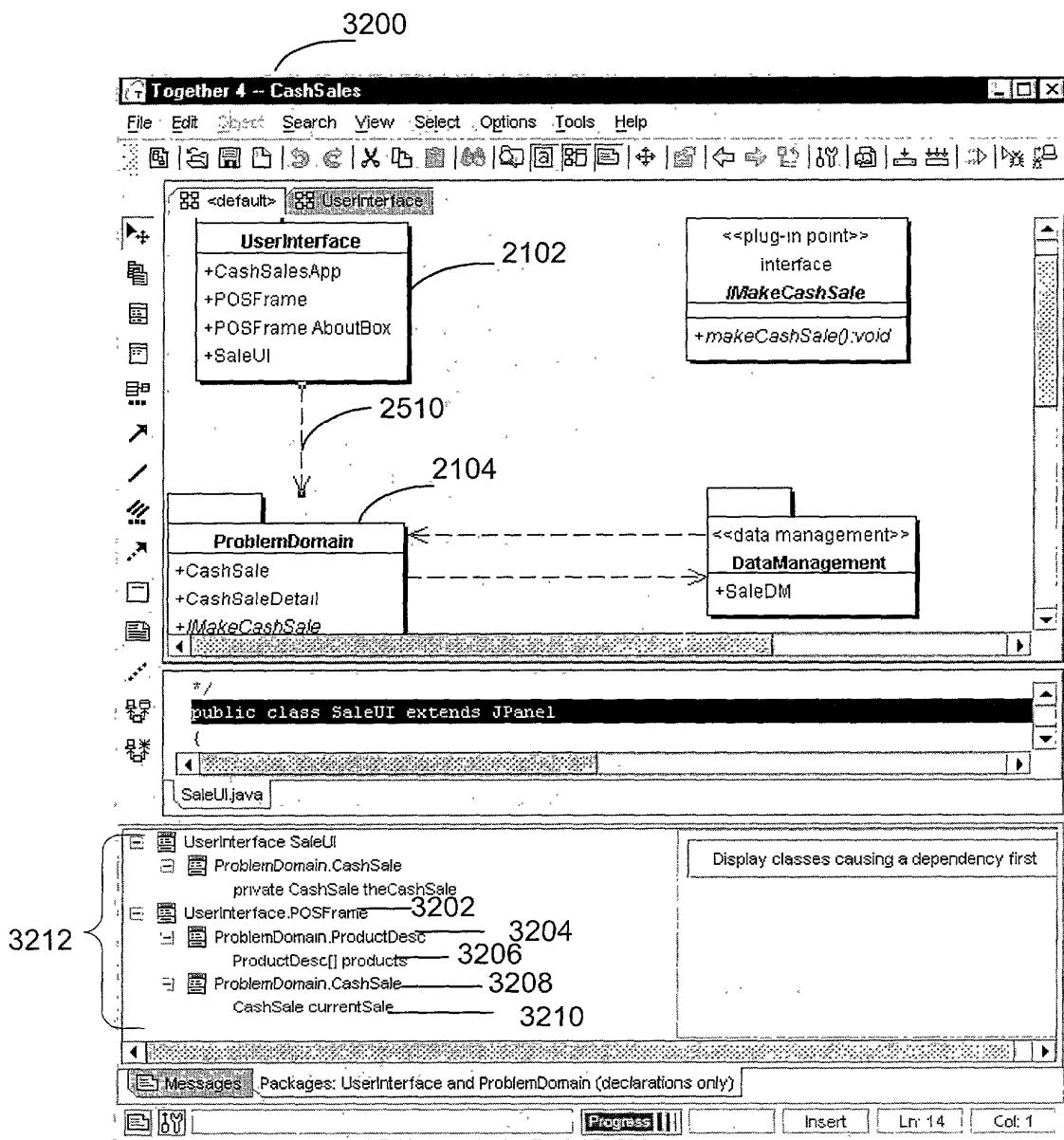


FIG. 33

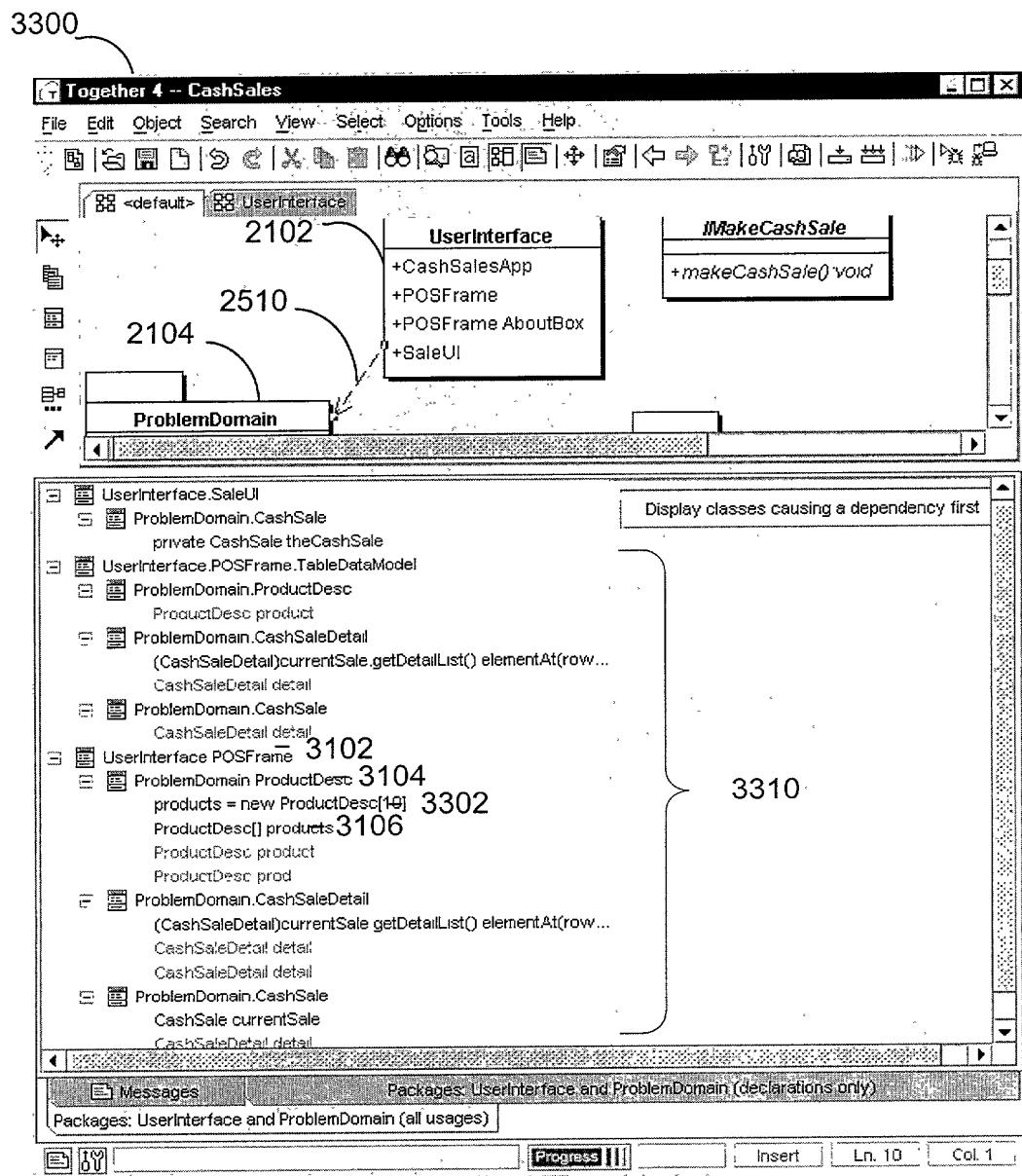


FIG. 34

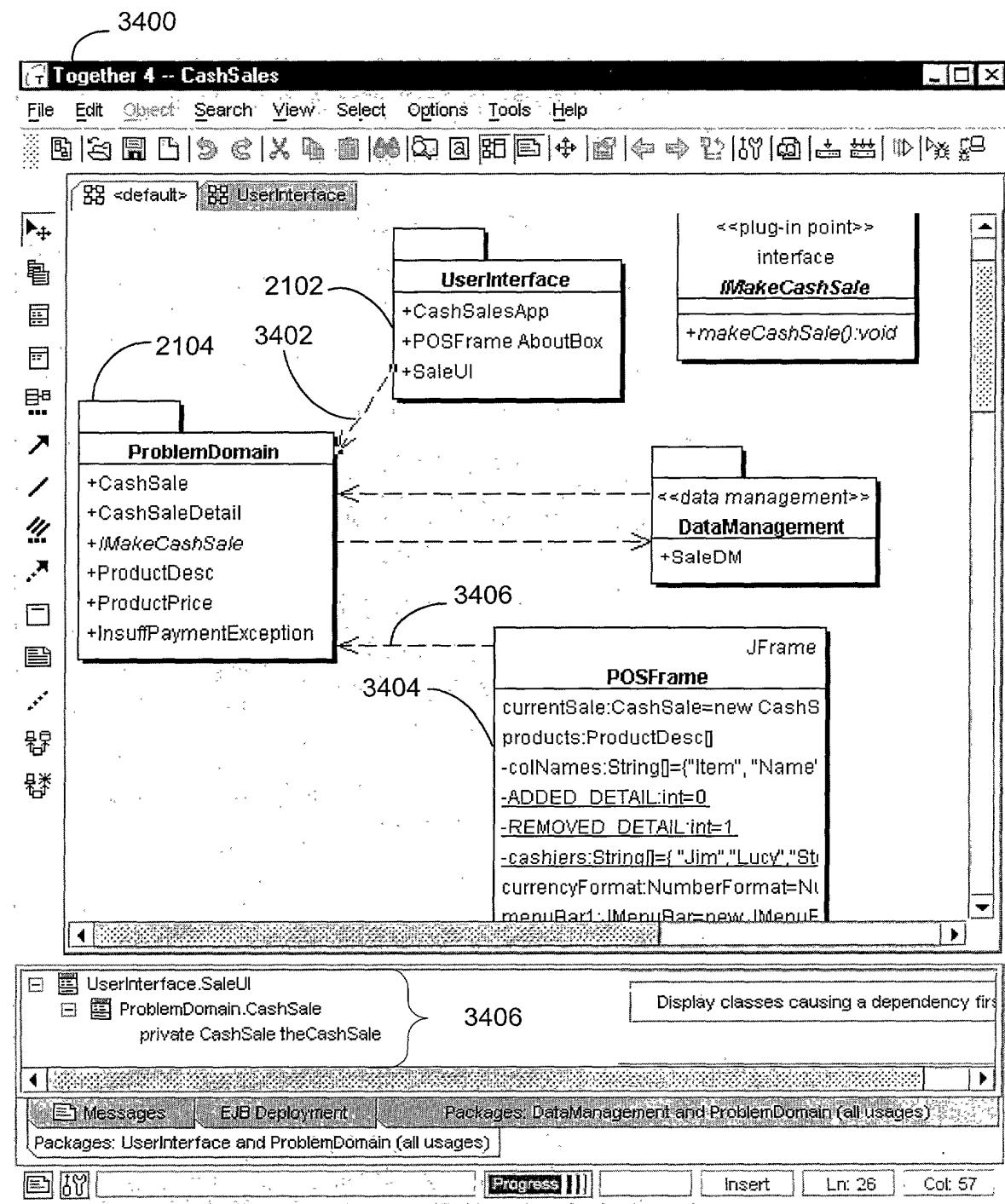


FIG. 35

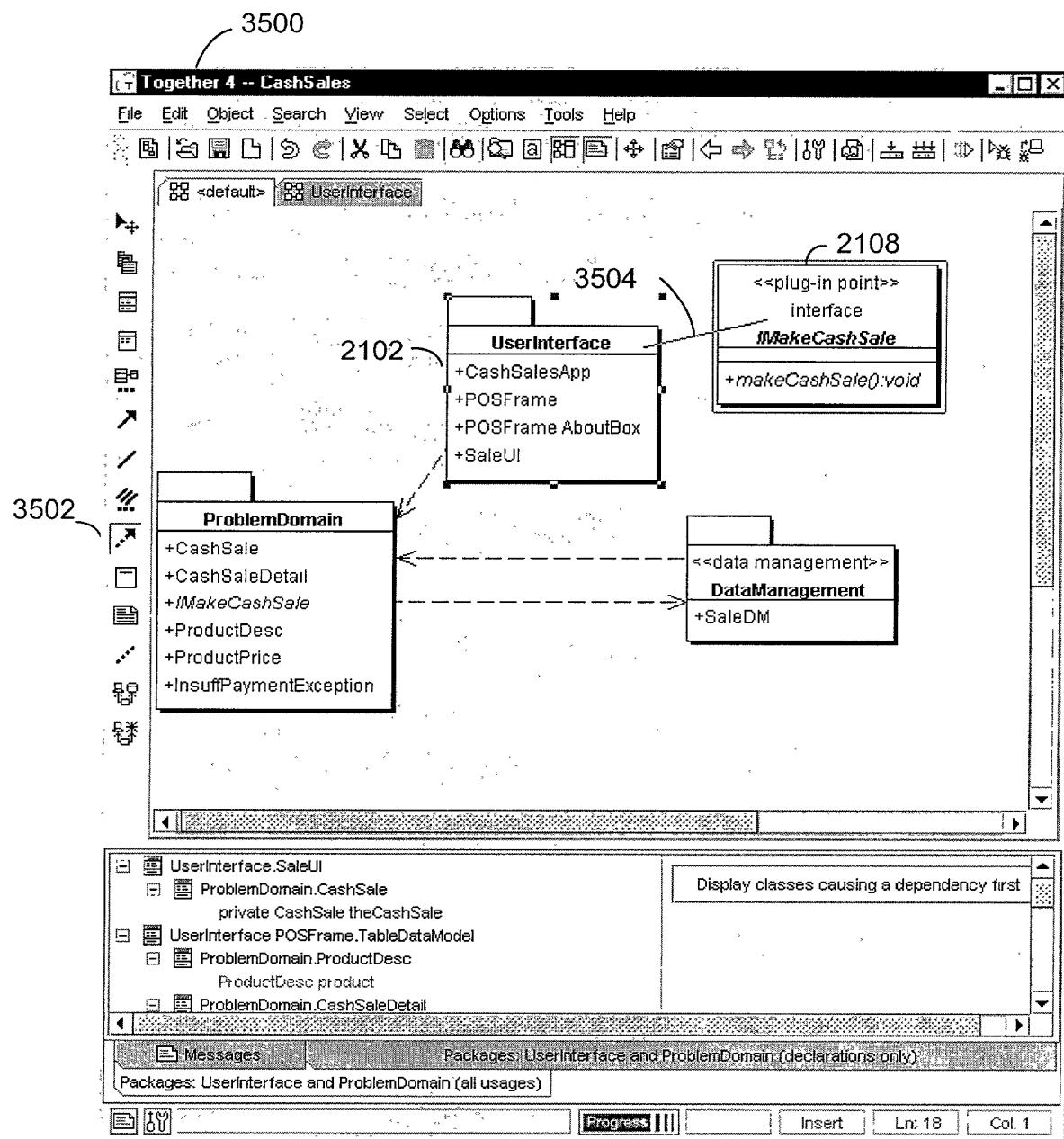


FIG. 36

